



PATENT
2565-0274P

IN THE U.S. PATENT AND TRADEMARK OFFICE

Applicant: Morio NAGATA, et al. Conf.: Unassigned
Appl. No.: 10/659,338 Group: Unassigned
Filed: November 1, 2003 Examiner: Unassigned
For: TRANSFORMATION APPARATUS, TRANSFORMATION
METHOD, TRANSFORMATION PROGRAMS, AND
COMPUTER READABLE RECORDING MEDIUM HAVING
THE TRANSFORMATION PROGRAM STORED THEREIN

LETTER

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

March 9, 2004

Sir:

Under the provisions of 35 U.S.C. § 119 and 37 C.F.R. § 1.55(a), the applicants hereby claim the right of priority based on the following application:

<u>Country</u>	<u>Application No.</u>	<u>Filed</u>
JAPAN	2002-278362	September 25, 2002

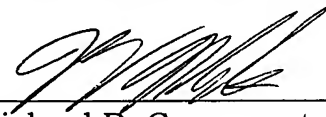
A certified copy of the above-noted application is attached hereto.

If necessary, the Commissioner is hereby authorized in this, concurrent, and future replies, to charge payment or credit any overpayment to Deposit Account No. 02-2448 for any additional fee required under 37 C.F.R. §§ 1.16 or 1.17; particularly, extension of time fees.

Respectfully submitted,

BIRCH, STEWART, KOLASCH & BIRCH, LLP

By


Michael R. Cammarata, #39,491

MRC/ĈJB:cb
2565-0274P

P.O. Box 747
Falls Church, VA 22040-0747
(703) 205-8000

Attachment(s)

日 本 国 特 許 庁
JAPAN PATENT OFFICE

10/659,338
f. 9-11-03
M. NAGATA, et al
Birch, Steward, et al
(703) 205-8000
1 of 1

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2 0 0 2 年 9 月 2 5 日

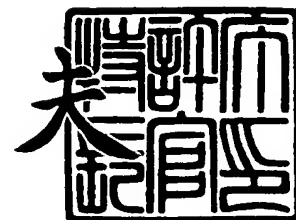
出 願 番 号
Application Number: 特 願 2 0 0 2 - 2 7 8 3 6 2
[ST. 10/C]: [J P 2 0 0 2 - 2 7 8 3 6 2]

出 願 人
Applicant(s): 学 校 法 人 慶 應 義 塾

2 0 0 3 年 9 月 2 6 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



出証番号 出証特 2 0 0 3 - 3 0 7 9 6 4 9

【書類名】 特許願

【整理番号】 000000345

【特記事項】 特許法第 3 0 条第 1 項の規定の適用を受けようとする特
許出願

【提出日】 平成14年 9月25日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/00

【発明者】

 【住所又は居所】 神奈川県横浜市港北区日吉 3 丁目 1 4 番 慶應義塾大学
 理工学部内

 【氏名】 永田 守男

【発明者】

 【住所又は居所】 神奈川県逗子市池子 3 丁目 1 3 番 2 8 号

 【氏名】 魚田 勝臣

【発明者】

 【住所又は居所】 神奈川県横浜市港北区日吉 3 丁目 1 4 番 慶應義塾大学
 理工学部内

 【氏名】 梶尾 義規

【特許出願人】

 【識別番号】 899000079

 【氏名又は名称】 学校法人 慶應義塾

【代理人】

 【識別番号】 100099461

 【弁理士】

 【氏名又は名称】 溝井 章司

【選任した代理人】

 【識別番号】 100118810

 【弁理士】

 【氏名又は名称】 小原 寿美子

【手数料の表示】

【予納台帳番号】 056177

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 変換装置及び変換方法及び変換プログラム及び変換プログラムを記録したコンピュータ読み取り可能な記録媒体

【特許請求の範囲】

【請求項 1】 バッチ処理をするプログラムをソースコードの形式で記憶する記憶部と、

上記記憶部が記憶したプログラムのソースコードを 1 以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する節判断部と、

上記節判断部が判断した各節の意味情報に基づいて上記記憶部が記憶したプログラムのソースコードからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソースコードとサーバ機器用の変換結果プログラムのソースコードとの 2 つからなる変換結果プログラムのソースコードに変換する抽出／変換部とを備える変換装置。

【請求項 2】 上記抽出／変換部は、上記 2 つの変換結果プログラムのソースコードを、オブジェクト指向プログラムのソースコードに変換する請求項 1 に記載された変換装置。

【請求項 3】 上記抽出／変換部は、所定のデータ構造と手続きをもつ複数のクラスに対応した複数のオブジェクト指向プログラムのテンプレートを生成し、上記 2 つの変換結果プログラムのソースコードから所定のデータ構造と手続きからなる情報を複数抽出し、抽出した各情報をテンプレートの対応する部分に適用することによって上記 2 つの変換結果プログラムのソースコードを複数のオブジェクト指向プログラムのソースコードに変換する請求項 2 に記載された変換装置。

【請求項 4】 バッチ処理をするプログラムをソースコードの形式で記憶する記憶部と、

上記記憶部が記憶したプログラムのソースコードを 1 以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する

節判断部と、

所定のデータ構造と手続きをもつ複数のクラスに対応した複数のオブジェクト指向プログラムのテンプレートを生成し、上記節判断部が判断した各節の意味情報に基づいて上記記憶部に記憶されたバッチ処理をするプログラムのソースコードから所定のデータ構造と手続きからなる情報を複数抽出し、抽出した各情報をテンプレートの対応する部分に適用することによって上記記憶部が記憶したプログラムのソースコードを複数のオブジェクト指向プログラムのソースコードに変換する変換装置。

【請求項 5】 上記変換装置は、さらに、

上記記憶部が記憶したプログラムのソースコードの役割をプログラムの意味情報として判断するプログラム判断部を備え、

上記抽出／変換部は、上記プログラム判断部が判断したプログラムの意味情報と上記節判断部が判断した各節の意味情報とに基づいてプログラムのソースコードからプログラムのソースコードを変換するための変換情報を抽出する請求項 1 又は請求項 4 のいずれかに記載された変換装置。

【請求項 6】 上記変換装置は、さらに、

上記記憶部が記憶したプログラムを構文解析する構文解析部を備え、

上記節判断部は、上記構文解析部によって構文解析されたプログラムに含まれる各節の意味情報を判断する請求項 1 または請求項 4 のいずれかに記載された変換装置。

【請求項 7】 上記変換装置は、バッチ処理をするコボルプログラムのソースコードを変換する請求項 1 または請求項 4 のいずれかに記載された変換装置。

【請求項 8】 バッチ処理をするプログラムをソースコードの形式で記憶し、

上記記憶したプログラムのソースコードを 1 以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断し、

上記判断した各節の意味情報に基づいて上記記憶したプログラムのソースコードからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソー

スコードとサーバ機器用の変換結果プログラムのソースコードとの2つからなる変換結果プログラムのソースコードに変換する変換方法。

【請求項9】 バッチ処理をするプログラムをソースコードの形式で記憶する処理と、

上記記憶したプログラムのソースコードを1以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する処理と、

上記判断した各節の意味情報に基づいて上記記憶したプログラムのソースコードからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソースコードとサーバ機器用の変換結果プログラムのソースコードとの2つからなる変換結果プログラムのソースコードに変換する処理とをコンピュータに実行させる変換プログラム。

【請求項10】 バッチ処理をするプログラムをソースコードの形式で記憶する処理と、

上記記憶したプログラムのソースコードを1以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する処理と、

上記判断した各節の意味情報に基づいて上記記憶したプログラムのソースコードからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソースコードとサーバ機器用の変換結果プログラムのソースコードとの2つからなる変換結果プログラムのソースコードに変換する処理とをコンピュータに実行させるための変換プログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、プログラム変換装置及びその方法に関するものである。

【0002】

【従来の技術】

従来からバッチ処理を行うコボルプログラムをCORBA、COMなどの技術

を用いて、ほぼ全体を人手によってクライアント／サーバ型のコボルプログラムに変換しようとする試みは存在した（例えば、非特許文献1参照）。

図57は一連のコボルプログラムをCORBA、COMなどの技術を用いて、クライアント／サーバ型コボルプログラムに変換する動作を示した従来図である。なお、ここで変換された結果は、クライアント／サーバ型ではあるが従来のコボルプログラムでありオブジェクト指向コボルプログラムではない。

このような試みでは、従来のコボル言語の手法を用いてコーディングされたコボルプログラム100をクライアント／サーバ型のプログラムに変換する。まず、人間が一連のコボルプログラム100のソースコードに記載された内容を理解する。そして、人間によってコボルプログラム100を内容的にまとまりのある個々のプログラムに分解する必要がある。また、CORBAやCOMなどと先程分解された個々のプログラムとを連携するためのインターフェースをインターフェース定義言語（IDL：INTERFACE DEFINITION LANGUAGE）によって記述する必要があるが、その作業にも人手が必要である。非特許文献1では、このIDL作成の一部のみを自動化している。

【0003】

【非特許文献1】

NEC Solutions、

Open COBOL Factory 21／

ObjectPartner Pro、

[平成14年7月12日検索]、インターネット<URL：

<http://www.sw.nec.co.jp/cced/ocf21/objptnpro/seihin.html>>

【0004】

【発明が解決しようとする課題】

このように、コボルプログラム100のソースコードから最終プログラム300のソースコードを生成するために行なわれる変換処理は、従来においては自動化されていないか、自動化されている部分があっても手動で行なわなければならない部分があり、人間の労力を軽減する変換方法が望まれていた。すなわち、従来、メインフレーム機器4000などに使われていたコボルプログラム100の

資源をクライアント／サーバシステム（分散システム）においても活用可能にするために、人的労力をかけない自動化された変換装置及び方法が望まれていた。

さらに、情報システムへの要請が集中型処理から分散型処理に変化している近年の状況を踏まえると、過去にコボルプログラムで構築した業務ロジックを再利用する方法がより一層求められていた。

【0005】

本発明は、既存のプログラムを新しい技術のソフトウェアに適した構造に変換することを目的とする。

【0006】

【課題を解決するための手段】

この発明に係る変換装置は、バッチ処理をするプログラムをソースコードの形式で記憶する記憶部と、

上記記憶部が記憶したプログラムのソースコードを1以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する節判断部と、

上記節判断部が判断した各節の意味情報に基づいて上記記憶部が記憶したプログラムのソースコードからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソースコードとサーバ機器用の変換結果プログラムのソースコードとの2つからなる変換結果プログラムのソースコードに変換する抽出／変換部とを備える。

【0007】

上記抽出／変換部は、上記2つの変換結果プログラムのソースコードを、オブジェクト指向プログラムのソースコードに変換することを特徴とする。

【0008】

上記抽出／変換部は、所定のデータ構造と手続きをもつ複数のクラスに対応した複数のオブジェクト指向プログラムのテンプレートを生成し、上記2つの変換結果プログラムのソースコードから所定のデータ構造と手続きからなる情報を複数抽出し、抽出した各情報をテンプレートの対応する部分に適用することによっ

て上記2つの変換結果プログラムのソースコードを複数のオブジェクト指向プログラムのソースコードに変換する。

【0009】

この発明に係る変換装置は、バッチ処理をするプログラムをソースコードの形式で記憶する記憶部と、

上記記憶部が記憶したプログラムのソースコードを1以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する節判断部と、

所定のデータ構造と手続きをもつ複数のクラスに対応した複数のオブジェクト指向プログラムのテンプレートを生成し、上記節判断部が判断した各節の意味情報に基づいて上記記憶部に記憶されたバッチ処理をするプログラムのソースコードから所定のデータ構造と手続きからなる情報を複数抽出し、抽出した各情報をテンプレートの対応する部分に適用することによって上記記憶部が記憶したプログラムのソースコードを複数のオブジェクト指向プログラムのソースコードに変換する。

【0010】

上記変換装置は、さらに、

上記記憶部が記憶したプログラムのソースコードの役割をプログラムの意味情報として判断するプログラム判断部を備え、

上記抽出／変換部は、上記プログラム判断部が判断したプログラムの意味情報と上記節判断部が判断した各節の意味情報とに基づいてプログラムのソースコードからプログラムのソースコードを変換するための変換情報を抽出する。

【0011】

上記変換装置は、さらに、

上記記憶部が記憶したプログラムを構文解析する構文解析部とを備え、

上記節判断部は、上記構文解析部によって構文解析されたプログラムに含まれる各節の意味情報を判断する。

【0012】

上記変換装置は、バッチ処理をするコボルプログラムのソースコードを変換す

る。

【0013】

この発明に係る変換方法は、バッチ処理をするプログラムをソースコードの形式で記憶し、

上記記憶したプログラムのソースコードを1以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断し、

上記判断した各節の意味情報に基づいて上記記憶したプログラムのソースコードからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソースコードとサーバ機器用の変換結果プログラムのソースコードとの2つからなる変換結果プログラムのソースコードに変換する。

【0014】

この発明に係る変換プログラムは、バッチ処理をするプログラムをソースコードの形式で記憶する処理と、

上記記憶したプログラムのソースコードを1以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する処理と、

上記判断した各節の意味情報に基づいて上記記憶したプログラムのソースコードからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソースコードとサーバ機器用の変換結果プログラムのソースコードとの2つからなる変換結果プログラムのソースコードに変換する処理とをコンピュータに実行させる。

【0015】

この発明に係るコンピュータに実行させるための変換プログラムを記録したコンピュータ読み取り可能な記録媒体は、バッチ処理をするプログラムをソースコードの形式で記憶する処理と、

上記記憶したプログラムのソースコードを1以上のまとまりある処理に区切り、区切った処理を節として節毎の役割を各節の意味情報として判断する処理と、

上記判断した各節の意味情報に基づいて上記記憶したプログラムのソースコー

ドからソースコード変換のための変換情報を抽出し、抽出した変換情報に基づいてプログラムのソースコードをクライアント機器用の変換結果プログラムのソースコードとサーバ機器用の変換結果プログラムのソースコードとの2つからなる変換結果プログラムのソースコードに変換する処理とを備える。

【 0 0 1 6 】

【発明の実施の形態】

実施の形態 1.

本実施の形態では、既存の資産であるコボルプログラム 1 0 0 のソースコードをまず中間プログラム 2 0 0 のソースコードに変換し、さらに、最終プログラム 3 0 0 のソースコードに変換する装置及び方法について説明する。

図 1 は、本実施の形態によるプログラム変換方法の一例を示す図である。コボルプログラム 1 0 0 はメインフレーム機器における集中型処理及びオフライン処理及び一括処理（バッチ処理）の環境で動作するコボル言語で記述されたプログラムである。このプログラムのソースコードを変換装置 A 1 0 0 0 によって中間プログラム 2 0 0 のソースコードに変換する。

中間プログラム 2 0 0 はクライアント／サーバシステムである分散処理及びオンライン処理及び一件別処理の環境において動作するオブジェクト指向プログラムである。中間プログラム 2 0 0 はインターフェースクラス 2 1 0 とファイルクラス 2 2 0 とから構成される。インターフェースクラス 2 1 0 はクライアント機器が処理するプログラムでありクライアント機器用の変換結果プログラムの一例である。ファイルクラス 2 2 0 はサーバ機器が処理するプログラムであり、サーバ機器用の変換結果プログラムの一例である。このように、コボルプログラム 1 0 0 をインターフェースクラス 2 1 0 とファイルクラス 2 2 0 に分解することによって、メインフレーム機器上で動作していたコボルプログラム 1 0 0 をネットワーク上につながれたクライアント機器とサーバ機器を連携させて処理するシステムに活用することができる。

なお、オブジェクトとは、外界の対象領域に存在するものについて、データ（属性）と手続き（メソッド）を一体化して表現したものをいう。また、クラスとは、オブジェクトの集合を抽象化した形で定義したものをいう。

【0017】

中間プログラム200のソースコードは、さらに、変換装置B2000によって最終プログラム300のソースコードに変換される。最終プログラム300は、WEBやVBやJava（登録商標）などに連携することが可能なオブジェクト指向プログラムの一例である。最終プログラム300は、中間プログラム200よりさらにオブジェクト指向性の高いプログラムである。また、最終プログラム300は、ビュークラス310、制御クラス320、モデルクラス330、セッションクラス340、エンティティクラス350から構成されるプログラムである。

このように、既存のコボルプログラム100をオブジェクト指向プログラムにまでソースコード変換することにより、インターネットなどのネットワーク上に構築された分散システム上で変換後のプログラムを動作させることができる。よって、既存のプログラムのさらなる有効活用が図れる。

【0018】

まず、変換装置A1000が、コボルプログラム100のソースコードを中間プログラム200のソースコードへ変換する方法について説明する。

図2は、コボルプログラム100によるオフライン一括処理を変換装置A1000によって中間プログラム200によるオンライン一件別処理に変換する一例を示す図である。

左側の一括処理では、メインフレーム機器が複数の入力データをチェック、ソートし、トランザクションファイルを生成した後、旧マスタファイルとマッチング、更新して、新マスタファイルを出力するとともに、必要ならばエラーリストを出力する。変換装置A1000は、このように、コボルプログラム100によるオフライン一括処理をクライアント／サーバシステムで行われるオンライン一件別処理に適合させるようにプログラムのソースコードを変換する。即ち、クライアント側では入力データの入力、チェックを行ない、トランザクションとしてサーバ機器に送る。サーバ機器ではクライアント機器から送られたトランザクションとマスタファイルをマッチングしてマスタファイルを更新するとともに、その結果をクライアント機器に伝える。

このように、変換装置 A 1 0 0 0 によってコボルプログラム 1 0 0 のソースコードを中間プログラム 2 0 0 のソースコードに変換することにより、オンラインの一件別処理が可能となる。

【 0 0 1 9 】

次に、上述したコボルプログラムでデータの追加や、更新削除などのトランザクションをオフライン一括処理を行うコボルプログラム 1 0 0 を中間プログラム 2 0 0 に変換することによってオンライン一件別処理を可能とする方法について、図 3 を用いてさらに詳細に説明する。

図 3 の左側はコボルプログラム 1 0 0 によるオフライン一括処理を示し、右側は中間プログラム 2 0 0 によるオンライン一件別処理を示す。

まず、入力・チェックプログラム 1 0 2 は、トランザクションファイル 1 0 1 のレコードを入力チェックし、チェック済みトランザクションファイル 1 0 3 を生成する。この処理の内容を記述した入力・チェックプログラム 1 0 2 のソースコードは、変換装置 A 1 0 0 0 によって、右側のインターフェースクラス 2 1 0 のプログラムのソースコードに変換される。

左側の一括処理で生成されたチェック済みトランザクションファイル 1 0 3 のレコードは右側のトランザクションレコード 2 0 3 に対応するものである。

次に、左側の一括処理では、チェック済みトランザクションファイル 1 0 3 をソートプログラム 1 0 4 によってソートし、チェック・ソート済みトランザクションファイル 1 0 5 を生成する。このソート処理は、右側の処理には不要である。なぜなら、右側の処理はバッチ処理ではなく一件別処理であるため、トランザクションレコード 2 0 3 は直接マッチング処理の対象レコードとなるからである。

次に、左側の一括処理では、チェック・ソート済みトランザクションファイル 1 0 5 と旧マスタファイル 1 0 6 をマッチングプログラム 1 0 7 によりマッチングする。その結果、新マスタファイル 1 0 8 を得る。この処理に対応して右側の一件別処理では、トランザクションレコード 2 0 3 とマスタファイル 2 0 6 をマッチングする（2 0 7）処理が行われる。このマッチング処理はファイルクラス 2 2 0 に記述され、サーバ機器によって実行される。

最後に、左側の一括処理では、結果出力プログラム 1 0 9 がマッチング処理か

ら得られた結果を出力する。この処理に対応して右側の一件別処理では、サーバ機器に結果が出力される（208）。

【0020】

次に、上記左側の一括処理を行うプログラムから右側の一件別処理を行うプログラムへ、そのソースコードを変換する変換装置A1000について説明する。

図4は、コボルプログラム100のソースコードを中間プログラム200のソースコードに変換する変換装置A1000の内部構成及び動作を示す図である。

変換装置A1000は変換前のコボルプログラム100を入力する入力部1100と、入力部1100が入力したプログラムを複数のプログラムに分割する分割部1200と、分割された各プログラムを構文解析する構文解析部1300と、構文解析したプログラムから各プログラムの内容を判断するプログラム判断部1400と、プログラム判断部1400が判断した各プログラム中の複数の節の内容を判断する節判断部1500と、節判断部1500が判断した節を用いてコボルプログラム100から中間プログラム200へ変換するために必要なデータを抽出し、抽出したデータを用いて中間プログラム200に変換する抽出／変換部1600と、変換した中間プログラム200を出力する出力部1700と、入力部1100が入力したプログラム等を記憶する記憶部1800とから構成される。なお、記憶部1800は必ずしも変換装置A1000の内部に存在する必要はなく、外部記憶装置を利用してもよい。

【0021】

次に、図4に示された各内部構成を用いてコボルプログラム100のソースコードを中間プログラム200のソースコードに変換する動作を説明する。

図5は、左から右に進行し、左から入力されたコボルプログラム100を変換して右の中間プログラム200を取得する動作を示した図である。

前述した通り、コボルプログラム100は、トランザクションファイル101を入力し、その内容をチェックする入力チェックプログラム102と、各チェック済みトランザクションレコードをソートするソートプログラム104と、チェック・ソート済みトランザクションファイル105と旧マスタファイル106をマッチングするマッチングプログラム107とマッチングした結果を出力する結

果出力プログラム 1 0 9 とから構成されている。ただし、入力チェックプログラム 1 0 2、ソートプログラム 1 0 4、マッチングプログラム 1 0 7、結果出力プログラム 1 0 9 のうちどれか一つが欠けていても構わない。

【 0 0 2 2 】

入力部 1 1 0 0 は、このコボルプログラム 1 0 0 を入力する。

次に、分割部 1 2 0 0 は、4 つのプログラムが 1 つにまとまった入力プログラムをそれぞれ 4 つのプログラムに分割する (S 1 2 0 0)。ただし、コボルプログラム 1 0 0 が一つのまとまったプログラムである場合には、分割部 1 2 0 0 は何も処理しない。

次に、構文解析部 1 3 0 0 は、分割された 4 つのプログラムに記載された各構文を解析する (S 1 3 0 0)。

次に、プログラム判断部 1 4 0 0 は、構文解析部 1 3 0 0 によって行なわれた構文解析に基づいて各プログラムの役割を判断する (S 1 4 0 0)。この判断の結果、入力チェックプログラム 1 0 2 は入力データの入力とチェック、ソートプログラム 1 0 4 は複数のレコードのソート、マッチングプログラム 1 0 7 はマスタファイルとのマッチング、結果出力プログラム 1 0 9 はマッチング結果の出力という役割を果たしていることが判断される。

次に、節判断部 1 5 0 0 がプログラム判断部 1 4 0 0 によって判断された各プログラム中の節の役割を判断する (S 1 5 0 0)。ここで、節とはプログラムを一又は複数のまとまりある処理に区切り、その区切った各処理をいう。

抽出／変換部 1 6 0 0 は、節判断部 1 5 0 0 によって判断された各節の役割からコボルプログラム 1 0 0 のソースコードを中間プログラム 2 0 0 のソースコードに変換するために必要なデータをコボルプログラム 1 0 0 のソースコードから抽出し、抽出したデータを中間プログラム 2 0 0 に適合するように変換する。

その結果、インターフェースクラス 2 1 0 とファイルクラス 2 2 0 とから構成される中間プログラム 2 0 0 が生成され、出力部 1 7 0 0 によって出力される。

【 0 0 2 3 】

次に、変換装置 A 1 0 0 0 の各部の動作について説明する。

まず分割部 1 2 0 0 の動作について説明する。

図6は、入力部1100が入力したコボルプログラム100の一例である。ここでは、前述したように、コボルプログラム100は4つのまとまりあるプログラムから構成されている。各プログラムの最終行には「END PROGRAM.」が存在する。

【0024】

図7は、図6で示したコボルプログラム100を複数のまとまりあるプログラムに分割する流れ図である。この流れ図は、PAD (PROBLEM ANALYSIS DIAGRAMS) によって記載されている。

ここでは、図6に示したプログラムに記述された見出し「IDENTIFICATION DIVISION.」から次の見出し「END PROGRAM.」までを一つのプログラムと判断し、それぞれ別個の出力ファイルに格納する。

すなわち、まず新しい出力ファイルを開き (S1201)、プログラムが終わるまで以下の処理を繰り返す (S1202)。

プログラムを一行読み (S1203)、「END PROGRAM.」があるかどうかを判断する (S1204)。

「END PROGRAM.」が存在しない場合には、読み込んだ1行を出力ファイルに書き出す (S1207)。

「END PROGRAM.」が存在する場合には、まとまるある1つのプログラムの最終と判断し、出力ファイルを閉じ (S1205)、新しい出力ファイルを開く (S1206)。

この処理を繰り返すことによって、たとえば、図6に示されたコボルプログラム100は4つのプログラム (入力・チェックプログラム102、ソートプログラム104、マッチングプログラム107、結果出力プログラム109) に分割される。

なお、図7においては見出し「END PROGRAM.」が存在するかを判断してプログラムを分割したが、入力ファイル名自体を用いて各々のファイル名から各まとまりあるプログラムに分割することも可能である。

また、分割部1200は必ずしも設ける必要はなく、構文解析部1300が直接入力されたプログラムに基づいて構文解析してもよい。

【 0 0 2 5 】

次に、分割部 1 2 0 0 によって分割された 4 個のプログラムの構文解析について説明する。構文解析部 1 3 0 0 は、分割された個々のプログラムを別個に構文解析する。その結果、各プログラムについてプログラムの階層構造を表現した構文解析木が得られる。

【 0 0 2 6 】

この構文解析木の各ノードには、プログラム中の命令に対する構文上の意味情報が付与されている。この構文解析木を作る操作はプログラム判断部 1 4 0 0 によって行なわれる。ここで、ノードとは、命令文の始まり及び終わりを境とし、プログラム中で意味を持つ最小の単位となる一連の語をまとめて格納したものをいう。

プログラム判断部 1 4 0 0 では各プログラム名の命名規則をもとに、入力チェックやマッチングなど、各プログラムの役割を判断し、判断した役割を意味情報として各プログラムに付与する。

ここで、プログラム名の命名規則について説明する。

プログラム中のプログラム名、ファイル、及びデータ項目については、予め各社内毎のコーディング規約によって命名規則が定められている。プログラム名については一連の処理に共通する名前と、入力、ソート、更新、結果出力の内どれかが識別できる名前が付いている。また、ファイル名にも各ファイルの役割を識別できる接辞が付与されている。

また、データ項目名には、ファイルのデータ項目ならばファイル名と同様にデータ項目の役割を識別できる接辞が付与されている。

これらプログラム中の命名規則に基づいて各プログラムに意味情報を付与することが可能である。

【 0 0 2 7 】

プログラム判断部 1 4 0 0 がプログラム名から各プログラムの役割を判断する流れ図を図 8 に示す。このフローは、プログラム判断部 1 4 0 0 によって実行される。

まず、プログラム判断部 1 4 0 0 は、構文解析部 1 3 0 0 によって解析された

各プログラムについて（S1401）、プログラム名を抽出する（S1402）。

プログラム名が入力チェックならば、プログラムに意味情報として「入力チェック」を付与する（S1404）。

プログラム名がソートならばプログラムに意味情報「ソート」を付与する（S1405）。

プログラム名がマッチング更新であるならば、プログラムに意味情報「マッチング更新」を付与する（S1406）。

プログラム名が結果出力ならばプログラムに意味情報「結果出力」を付与する（S1407）。

【0028】

次に、節判断部1500が各プログラムの構文解析木の各節のノードを1つずつ抽出し、そのノードへ意味情報を付与する処理について説明する。

各プログラムは、前述したように、プログラム中のまとまりある処理である節から構成されている。

図9は、入力チェックプログラム102の各節の構造を示している。四角で囲まれた各処理が節に相当する。図9では、左の節が右の節を呼び出す構造になっている。

主処理（S129）はトランザクション入力処理（S138）と変換コントロール処理（S135）を呼び出している。トランザクション入力処理は、トランザクションファイル101からレコードを読み込む処理であり、変換コントロール処理は変換コントロール処理が呼び出す各処理（変換処理、トランザクション入力処理）を繰り返す制御処理である。

変換コントロール処理（S135）で行なわれる繰り返し処理は、変換処理（S139）とトランザクション入力処理（S138）を呼び出すことにより行なわれる。変換処理が行なわれた後には、チェック済トランザクション出力処理（S141）の節によってチェック済トランザクションレコードをファイルに書き出す処理を行う。なお、S139の変換処理では、トランザクションレコードをチェックし、正しいレコードを出力側に転記する処理を行なっている。

このようにして、節判断部1500は、図9で示す入力・チェックプログラム102の節の構造から各節の役割を判断する。

【0029】

節判断部1500が図9に示した各節の役割を判断する流れ図を図10に示す。

まず、節判断部1500は節のノードを一つずつ抽出し(S1501)、OPEN文又はREAD文又はWRITE文を含むかを判断する(S1502)。

節判断部1500は、節に対応する構文解析木のノードにOPEN文を含むと判断されている場合には節の役割として意味情報「主処理」を付与する(S1503)。READ文を含む場合には節の役割として意味情報「トランザクション入力」を付与する(S1504)。WRITE文を含む場合には節の役割の意味情報として「チェック済トランザクション出力」を付与する(S1505)。

次に、節判断部1500は、他の節に対してチェック済出力をPERFORM文で読み出しているかを判断し(S1507)、呼び出している場合にはその節の役割として意味情報「変換処理」を付与する(S1508)。

節判断部1500は、他の節に対して変換処理をPERFORM文で呼び出している判断される場合には(S1510)、その節の役割として意味情報「変換処理コントロール」を付与する(S1511)。このようにして、各節の役割として意味情報を自動的に付与することができる。

【0030】

次に、節判断部1500がマッチングプログラム107に対して節の判断と意味情報の付与を行う動作を説明する。

図11は、マッチングプログラム107の節の構造を示している。図9と同様に四角で囲まれた処理は節を表し、左の処理が右の処理を呼び出す関係となっている。主処理(S157)はトランザクションを入力し(S165)、旧マスタファイルを入力し(S166)、これらのデータを元に更新コントロールに基づいて更新処理を繰り返す。すなわち、更新コントロールでは、トランザクションキーとマスタキーを照合する合致処理を行ない(S159)、また次のマスタレコードを準備するマスタ処理(S169)と次のトランザクションレコードを準

備するトランザクション処理（S 1 6 0）に基づいて合致処理（S 1 5 9）を繰り返す。その結果は新マスタファイルに出力される。このようにマッチングプログラム 1 0 7 は図 1 1 に示す節の構造を持っているので、節判断部 1 5 0 0 はこれらの節の構造から各節の役割を判断する。

【 0 0 3 1 】

節判断部 1 5 0 0 が図 1 0 に示した各節の役割を判断する流れ図を図 1 2 に示す。

まず、節判断部 1 5 0 0 は、節が対応する構文解析木のノードを一つずつ抽出し（S 1 5 2 1）、OPEN 文、トランザクションの READ 文、旧マスタファイルの READ 文、新マスタファイルの READ 文があるかどうかを判断する（S 1 5 2 2）。OPEN 文がある場合には、その節の役割として、意味情報「主処理」を付与する（S 1 5 2 3）。トランザクションの READ 文がある場合にはその節に意味情報「トランザクション入力」を付与する（S 1 5 2 4）。旧マスタファイルの READ 文がある場合には、その節に意味情報「旧マスタファイル入力」を付与する（S 1 5 2 5）。新マスタファイルの READ 文がある場合にはその節に意味情報「新マスタファイル出力」を付与する（S 1 5 2 6）。

また、節判断部 1 5 0 0 は、残りの節について、前述したプログラム中の命名規則に従ってプログラム名、ファイル名、データ項目名を用いてどのレコード定義がトランザクションファイル、新マスタファイル、旧マスタファイルのいずれかのレコードかを判断することができる（S 1 5 2 8）。

【 0 0 3 2 】

以上、節判断部 1 5 0 0 が入力・チェックプログラム 1 0 2 とマッチングプログラム 1 0 7 について節の判断を行なった後、抽出／変換部 1 6 0 0 が各節の判断に基づいて中間プログラム 2 0 0 を生成するのに必要なデータを抽出し、変換する。次に、抽出／変換部 1 6 0 0 で行なわれる抽出／変換処理について説明する。

まず、前処理として中間プログラム 2 0 0 に予め各クラス中のメソッドを作り込む作業が必要である。ここで、メソッドとは具体的処理方法（手段）をいう。

抽出／変換部 1 6 0 0 は、この前処理として中間プログラム 2 0 0 を構成する

インターフェースクラス 2 1 0 及びファイルクラス 2 2 0 の 2 つ構造を作り、各クラスにファイル名情報をつけておく。

また、上記 2 つのクラスには各々次の段落に示すメソッドを作り、中身は空にしておく。各メソッドには意味情報を付与しておき、後で抽出及び変換する際に各メソッドを特定できるようにする。

【 0 0 3 3 】

図 1 3 は、入力チェックプログラム 1 0 2 と結果出力プログラム 1 0 9 とインターフェースクラス 2 1 0 の対応図である。

図 1 4 は、マッチングプログラム 1 0 7 とソートプログラム 1 0 4 とファイルクラス 2 2 0 の対応図である。

図 1 3 に示すように、抽出／変換部 1 6 0 0 は予め、インターフェースクラス 2 1 0 に画面表示入力メソッド（メソッド：displayScreen）と UI メインメソッド（メソッド：uiMain）と入力チェックメソッド（メソッド：changeModel）を作っておく。

また、抽出／変換部 1 6 0 0 は予め、ファイルクラス 2 2 0 にマッチング更新メソッド（メソッド：updateRecord）を作っておく。

【 0 0 3 4 】

このような前処理を行なった後、図 1 3 に示すように、抽出／変換部 1 6 0 0 は、入力チェックプログラム 1 0 2 と結果出力プログラム 1 0 9 からデータを抽出、変換し、クライアント機器 5 0 0 0 用の変換結果プログラムとしてインターフェースクラス 2 1 0 を生成する抽出変換処理を行う。また、図 1 4 に示すように、抽出／変換部 1 6 0 0 はマッチングプログラム 1 0 7 とソートプログラム 1 0 4 からデータを抽出、変換し、サーバ機器 6 0 0 0 用の変換結果プログラムとしてファイルクラス 2 2 0 を生成する抽出変換処理を行う。

以下に、上記抽出／変換部 1 6 0 0 が行う抽出変換処理を説明する。

【 0 0 3 5 】

・入力チェックプログラム 1 0 2 と結果出力プログラム 1 0 9 からインターフェースクラス 2 1 0 への抽出変換処理

図 1 3 に示すように、抽出／変換部 1 6 0 0 では入力チェックプログラム 1 0

2 と結果出力プログラム 109 とからインターフェースクラス 210 を生成するために対応付けを行う。

抽出／変換部 1600 は入力チェックプログラム 102 から入力チェックのロジックを抽出し、インターフェースクラス 210 に対応付ける。また、抽出／変換部 1600 は結果出力プログラム 109 から伝票定義を抽出し、インターフェースクラス 210 に対応付ける。

【0036】

抽出／変換部 1600 は、次に示す 3 つのロジックの対応付けに基づいて抽出変換処理を行う。

第 1 に入力チェックプログラム 102 では、トランザクションファイル 101 からレコードを読み込むロジックが存在していたが、インターフェースクラス 210 では画面からデータを入力してレコード形式に格納するロジックに変換する必要がある。すなわち、トランザクションファイル 101 からレコードを読み込むロジックは不要となる。したがって、第 1 の対応付けから入力チェックプログラム 102 に存在していたトランザクションファイル 101 からレコードを読み込むロジックを無視する。

第 2 に入力チェックプログラム 102 では読み込んだ各レコードのデータをチェックするロジックが存在するが、インターフェースクラス 210 においてもこのロジックを継承する必要がある。

第 3 に入力チェックプログラム 102 ではデータが正しければレコードをチェック済トランザクションファイル 103 に書き出すロジックが存在していたが、インターフェースクラス 210 においてはデータが正しければレコードをファイルクラス 220 に送るロジックに変換する必要がある。すなわち、データが正しければレコードをチェック済トランザクションファイル 103 に書き出すロジックは不要となる。したがって、第 3 の対応付けから入力チェックプログラム 102 に存在していたチェック済トランザクションファイル 103 の定義部分とチェック済トランザクションファイル 103 に書き出すロジックを無視し、レコードをチェックして正しければファイルクラス 220 のメソッドに送るようにするロジックを加える。

また、一括処理から一件別処理へ処理方法が変更されるので、入力チェックプログラム 102 に存在していた次のレコードを準備する繰り返しのロジックを無視する。

【0037】

これらの対応付けに基づいたプログラムの抽出、変換の詳細を図 15 及び図 16 に示す。図 15 は入力チェックプログラム 102 からインターフェースクラス 210 のプログラムへの抽出及び変換の詳細を説明する図である。図 16 は結果出力プログラム 109 からインターフェースクラス 210 のプログラムへの抽出及び変換を説明する図である。図 15 及び図 16 に示す位置番号は図 13 及び図 14 に示す対応付けの番号に対応している。すなわち、入力チェックプログラム 102 の見出し部にかかれたプログラム名からインターフェースクラス 210 の見出し部のクラス名及び環境部のファイルクラス 220 へのリポジトリ指定及びクラス終わり見出しの抽出変換を行うが、その詳細については図 15 の位置番号 (1-1) に示されている。

【0038】

この抽出変換の詳細について、図 17 と図 18 を用いて説明する。

図 17 は、入力チェックプログラム 102 の一部を示す図である。図 18 は、インターフェースクラス 210 の一部を示す図である。

図 17 の位置番号 (1-1) で示されたステップから図 18 の (1-1) で示されたステップが抽出変換される。

さらに、具体的に説明すると、図 17 の見出し部 (S121) 中の PROGRAM-ID (S122) の行から「在庫マスタ修正-入力チェック」を抽出し、図 18 のインターフェースクラス 210 中の見出し部 (S221) のクラス ID 「在庫マスタ修正 UI」として変換する。

また、インターフェースクラス 210 内の環境部 (S222) に記載されたファイルクラス 220 に対するリポジトリ指定の内部名として、入力チェックプログラム 102 から抽出した「在庫マスタ修正」にファイルクラス名の接辞を付加するとともに、外部名については各会社毎のコーディング規約中の命名規則に従ってファイルクラス 220 のファイル名を挿入する。

クラスの終わりを示す見出しについては後述する。

【0039】

次に、図17で示す入力チェックプログラム102のデータ部(S123)のトランザクションファイル定義に基づき、インターフェースクラス210のクラス変数を抽出変換する。具体的には図17に示された位置番号(1-2)から抽出し、図18のデータ部(S223)に示された位置番号(1-2)に変換する。このように入力チェックプログラム102のトランザクションファイルの接辞がついたレコードをインターフェースクラス210のクラス変数として記載し、画面からの入力をレコードの形式にしてファイルクラス220に送ることが可能となる。

【0040】

次に、入力・チェックプログラム102中の手続き部からインターフェースクラス210の入力チェックメソッドを作成するための抽出変換処理について説明する。

図19は、入力チェックプログラム102の手続き部を示している。図20は、インターフェースクラス210の入力チェックメソッドの手続き部とクラスの終わりの見出し部分を示している。

図13に示すように、入力チェックプログラム102の手続き部(S128)はインターフェースクラス210の入力チェックメソッド「changeModel」(S227)の手続き部に変換される。各変換処理についての詳細は、図15の位置番号(1-4)から(1-8)に示されているが、この変換処理について図19と図20を用いて実際の具体的抽出変換を説明する。

【0041】

図19の手続き部(S128)中の主処理(S129)に記載された主処理節(S130からS134)を抽出し、この抽出した要素から図20に示す入力チェックメソッドの手続き部の主処理(S228)の処理内容を生成する変換処理を説明する。なお、抽出時の主処理部分の特定の仕方については前述したとおりであるので、省略する。

実際の構造及び構文上の変換については、まず、図19で示すファイルのOP

EN文(S130)及びファイルのCLOSE文(S133)を無視する。また、変換処理コントロールへのPERFORM文(S132)から繰り返しの指定(UNTIL指定)を無視する。ファイルのOPEN文及びCLOSE文を無視するのは、インターフェースクラス210では入力画面で行ない、チェック済レコードはファイルクラス220に出力するために不要となるためである。また、繰り返しの指定を無視するのは、中間プログラムではレコードを1件しか処理しないためである(一件別処理)。

次に、STOP RUN.(S134)を中間プログラムの構文上必要なEXIT METHOD.に変換する。これは図20のS238に示されている。

【0042】

次に、位置番号(1-5)に示される処理の変換について説明する。

図19において、変換処理コントロール(S135)に記載された具体的内容(S136、S137)を抽出し、トランザクション入力へのPERFORM文(S137)を消去する。これは、中間プログラムではレコードを1件しか処理しないので、次のトランザクションレコードを入力する必要があるためである。その結果、図20にはS231及びS232のみが抽出される。

【0043】

次に、位置番号(1-6)の抽出変換について説明する。

図19のS138で示すトランザクションファイルのREAD文を無視し、処理を何も行わないことを表すCONTINUE文に置き換える。中間プログラムでは画面からの入力データが直ちにトランザクションレコードに入るため、READ文は不要となるためである。その結果、トランザクション入力処理は図20のS234で示す部分のように変換される。

【0044】

次に位置番号(1-7)の変換処理についての抽出変換を説明する。

図19のS139で示す変換処理の内、S142のMOVE文、すなわち、トランザクションレコードからチェック済トランザクションレコードへレコードを

移動させるための文を無視する。中間プログラムによれば、トランザクションレコードをチェックしてデータが正しければそのままファイルクラス 220 にそのデータを送るためである。その結果、変換処理には図 20 の S 235 で示す部分のみが抽出される。

【0045】

次に、位置番号（1-8）の抽出変換処理について説明する。

ここでは、図 19 の S 141 に示すトランザクション出力処理の抽出変換処理が行なわれる。具体的にはチェック済トランザクションファイルへの WRITE 文（S 143）をファイルクラス 220 のマッチング更新メソッドへの INVOKE 文に置き換える。中間処理プログラムにおいては、チェックしたレコードはファイルクラス 220 のマッチング更新メソッドへの引数となるからである。このようにして、抽出変換した結果を図 20 の S 236 に示す。

なお、S 237 は位置番号（1-1）に対応してクラスの終わりの見出しを抽出変換した変換後のステップを示している。

【0046】

以上のようにして抽出／変換部 1600 はコボルプログラム 100 に記載された入力チェックプログラム 102 のソースコードから中間プログラム 200 を構成するインターフェースクラス 210 のソースコードを自動的に抽出変換する。

実際の入力チェックプログラム 102 のソースコードを図 21 に示す。また、入力チェックプログラム 102 のソースコードから抽出／変換部 1600 によって抽出、変換されたインターフェースクラス 210 のソースコードを図 22、図 23 に示す。すなわち、図 22 の第 1 行であるステップ 1（000001）から図 23 の最終行であるステップ 100（000100）までのソースコードが抽出、変換されたインターフェースクラス 210 のソースコードである。

図 21、図 22、図 23 に示すプログラム中、各位置番号はどのように変換がなされたかを説明するためのものであり、実際のプログラムに記載する必要はない。また、上記において説明されていない位置番号については後述する。

【0047】

次に、図 16 に示す結果出力プログラム 109 からインターフェースクラス 2

10への抽出、変換の具体的動作を説明する。図24、図25に結果出力プログラム109のプログラムを記載する。図24の第1行であるステップ1（000001）から図25の最終行であるステップ84（000084）までのソースコードが結果出力プログラム109のソースコードである。

上記結果出力プログラム109に記載された位置番号（1-25）から伝票定義を抽出し、図18の位置番号（1-25）に示す画面表示メソッドの画面定義に変換する。この場合、伝票の行位置、桁位置を画面の行位置、桁位置に対応付ける必要がある。

また、上記結果出力プログラム109の位置番号（1-25）に示す伝票項目の「SOURCE指定」を図18の位置番号（1-25）に示す画面項目の「TO指定」に置き換えることが必要となる。

さらに、トランザクションレコードに含まれているがマスタレコードに含まれないデータ項目をもとにそれらの項目について画面項目を生成する。結果出力プログラム109はマスタレコードの項目を出力しているが、インターフェースクラス210の画面表示メソッドでは画面からトランザクションの項目を入力するためである。

以下に、上記抽出／変換部1600がマッチングプログラム107とソートプログラム104からファイルクラス220へ変換する情報を抽出する抽出変換処理を図14を用いて説明する。

【0048】

・マッチングプログラム107とソートプログラム104からファイルクラス220への抽出変換処理

次に、マッチングプログラム107とソートプログラム104とに基づいて中間プログラムのファイルクラス220を抽出、変換する動作について説明する。前述したように、図14は左側に示すマッチングプログラム107とソートプログラム104から必要なデータを抽出し、抽出したデータに基づいてマッチングプログラム107またはソートプログラム104を変換し、右側に示すファイルクラス220を生成するための各対応付けを示している。

【0049】

マッチングプログラム 1 0 7 とソートプログラム 1 0 4 とから中間プログラム 2 0 0 を生成するための各対応についての基本的方針を説明する。

まず、マッチングプログラム 1 0 7 からはマッチング更新のロジックを抽出する。また、ソートプログラム 1 0 4 からはレコードキーの情報を抽出する。

マッチング更新のロジックは次の 3 つの対応付けにより変換する。

第 1 はマッチングプログラム 1 0 7 ではチェック・ソート済みトランザクションファイル 1 0 5 からレコードを読み込んでいたが、ファイルクラス 2 2 0 ではトランザクションレコードを手続きの引数として受け取るように変換する必要がある。

第 2 にマッチングプログラム 1 0 7 ではマスタレコードとチェック・ソート済みトランザクションファイル 1 0 5 のレコードをマッチングするロジックが存在していたが、ファイルクラス 2 2 0 においてもこのロジックが必要となる。

第 3 にマッチングプログラム 1 0 7 ではマッチングした結果、データが正しければ処理区分に応じて、新マスタレコードを更新し、新マスタファイルに書き出す動作を行っていたが、ファイルクラス 2 2 0 ではデータが正しければ処理区分に応じてマスタレコードを更新し、マスタファイルを書き換えるという処理に変換する必要がある。

【 0 0 5 0 】

上記第 1 ～第 3 の対応付けに従い、抽出／変換部 1 6 0 0 は、マッチングプログラム 1 0 7 で存在していたトランザクションファイルからレコードを読み込むロジックを無視し、トランザクションレコードを引数として定義する。

また、抽出／変換部 1 6 0 0 は、マッチングプログラム 1 0 7 で存在していたマスタファイルの定義のどちらか一方を無視し（この例では新マスタファイルを無視する）マスタファイルへの読み書きは無視しなかった残る一方のファイルのみに行うようにロジックを変換する。

また、抽出／変換部 1 6 0 0 は、マスタファイルを索引編成ファイルにする。

抽出／変換部 1 6 0 0 は、マスタレコードを書き出す命令を、追加（W R I T E）、更新（R E W R I T E）、削除（D E L E T E）にする。

トランザクションレコード、マスタレコードとも、ファイルクラス 2 2 0 にお

いては一件のみ処理するので、抽出／変換部 1 6 0 0 は次のレコードを準備する
繰り返しのロジックを無視する。

以上の方針に基づき、抽出／変換部 1 6 0 0 は変換前のマッチングプログラム
1 0 7 とソートプログラム 1 0 4 とから必要な情報を抽出、変換し、ファイルク
ラス 2 2 0 のソースコードを自動生成する。

【 0 0 5 1 】

上記抽出／変換部 1 6 0 0 による抽出、変換の詳細を図 2 6 ～図 2 8 に示す。
図 2 6、図 2 7 は、マッチングプログラム 1 0 7 からファイルクラス 2 2 0 への
抽出、変換の詳細を示す図である。図 2 8 は、ソートプログラム 1 0 4 からファ
イルクラス 2 2 0 への抽出、変換を示す図である。図 2 6 ～図 2 8 に示された位
置番号は図 1 4 の対応付けに示された位置番号と一致している。このように、抽
出／変換部 1 6 0 0 が自動的に必要な情報を抽出変換し、ファイルクラス 2 2 0
が生成される。

【 0 0 5 2 】

抽出／変換部 1 6 0 0 は、コボルプログラム 1 0 0 にはないが中間プログラム
2 0 0 で必要となる要素を新規に生成して追加する。すなわち、インターフェー
スクラス 2 1 0 に次の 3 つの要素を追加する。第 1 の要素は、画面からの入力
が終了かを判定するフラグ項目である。第 2 の要素は、画面表示メソッドや、入力
チェックメソッドを呼び出すメイン手続である。第 3 の要素は画面表示メソッド
内の表示及び入力受付命令である。

【 0 0 5 3 】

以上に説明したマッチングプログラム 1 0 7 のソースコードを図 2 9 ～図 3 1
に示す。また、ソートプログラム 1 0 4 のソースコードを図 3 2 に示す。さらに
、ファイルクラス 2 2 0 のプログラムを図 3 3、図 3 4 に示す。

プログラム中の位置情報は図 1 4、図 2 6 及び図 2 7 及び図 2 8 との対応を明
確に示すために記載しているが、実際のプログラムには不要である。

【 0 0 5 4 】

本実施の形態の変換装置 A 1 0 0 0 によれば、集中型処理に適合したコボルプ
ログラム 1 0 0 のソースコードから分散型処理に適合した中間プログラム 2 0 0

のソースコードへの自動変換が可能となる。従って、集中型処理に用いられていたプログラム資産を分散型処理においても使用することができるため、プログラム資産の有効活用が図れる。

【0055】

また、本発明の実施の形態によれば、旧来コーディングされたプログラム中の業務ロジックを再利用することができる。

【0056】

また、プログラムの変換作業は変換装置A1000によって自動的に行われるために、人的労力を必要としないため、労力の低減を図ることができる。

【0057】

また、企業などの組織において、集中型処理から分散型処理へシステムを移行する場合に、集中型処理に用いていたプログラムを分散型処理においても有効に使用できるため、新たにプログラムを開発する必要を最小限に抑えることができ、人的労力の軽減、システム構築期間の短縮化及びシステム構築費用の低減を図ることができる。

【0058】

また、手続き型プログラムからオブジェクト指向プログラムに変換することによって、プログラム間のインターフェースをより明確にできる。

【0059】

また、手続き型プログラムからオブジェクト指向プログラムに変換することによって、オブジェクト内部の仕様変更が外部に及ばないようなプログラムが可能になり、ソースコードを再利用しやすくなる。

【0060】

次に、図1に記載された中間プログラム200から最終プログラム300を生成する変換装置B2000について説明する。この変換装置B2000により、中間プログラム200をよりオブジェクト指向的な、個別の役割に特化したクラスからなる最終プログラム300に変換することができる。

まず、中間プログラム200から最終プログラム300へプログラム変換を行う変換装置B2000の内部構成について説明する。

図 35 は変換装置 B 2000 の内部構成図である。

入力部 2100 は変換装置 A 1000 によって変換された中間プログラム 200 を入力する。

次に、抽出／変換部 2200 が入力部 2100 によって入力された中間プログラム 200 から必要な情報を抽出し、変換する。出力部 2300 は抽出／変換部 2200 によって抽出変換されたプログラムを最終プログラム 300 として出力する。入力部 2100、抽出／変換部 2200、出力部 2300 は必要に応じて記憶部 2400 に情報を記憶することができる。たとえば、入力部 2100 は入力した中間プログラム 200 を記憶部 2400 に記憶することができる。なお、記憶部 2400 は必ずしも変換装置 B 2000 の内部に存在する必要はなく、外部記憶装置を利用してもよい。

【0061】

次に、変換装置 B 2000 により中間プログラム 200 を最終プログラム 300 に変換する場合において、各データ処理の変換の流れを説明する。

図 36 は、変換装置 B 2000 によってデータ処理の方法がどのように変換されたかを示す図である。

図の左側は中間プログラム 200 によるオンライン一件別処理を示している。図の右側は、最終プログラム 300 による、中間プログラム 200 よりさらにオブジェクト指向的なオンライン一件別処理を示している。

左側の処理を行う中間プログラム 200 はインターフェースクラス 210 とファイルクラス 220 とを持つ。左側のデータの流については図 3 の説明で行なったので、ここでは省略する。

【0062】

右側の最終プログラム 300 は 5 つのクラスから成り立っている。具体的にはインターフェースクラス 210 からビュークラス 310、制御クラス 320 及びモデルクラス 330 の 3 つのクラスが生成される。また、ファイルクラス 220 からセッションクラス 340 及びエンティティクラス 350 の 2 つのクラスが生成される。

ビュークラス 310 と制御クラス 320 とモデルクラス 330 とはクライアント

ト機器 5000 側のクラスである。このうち、ビュークラス 310 は画面表示と入力の受付を行うクラスである。モデルクラス 330 はデータのモデル（属性など）を管理するクラスである。制御クラス 320 はビュークラス 310 によって管理される画面と、モデルクラス 330 によって管理されるデータモデルの制御を行うクラスである。

【0063】

セッションクラス 340 とエンティティクラス 350 とはサーバ機器 6000 側のクラスである。セッションクラス 340 はマスタレコードとトランザクションレコードの照合を行うクラスである。エンティティクラス 350 はセッションクラス 340 によって行なわれた照合の結果を記憶媒体へ書き込むことを管理するクラスである。

【0064】

このような各クラスの制御に従って、トランザクションレコード 303 をクライアント機器 5000 側からサーバ機器 6000 側に受け渡し、よりオブジェクト指向的なオンライン一件別処理が可能となる。

【0065】

次に最終プログラム 300 を生成するために抽出／変換部 2200 が行う動作について説明する。

まず、抽出／変換部 2200 は最終プログラム 300 に予め作り込む要素を生成する。生成した要素は、図 36 に示した 5 つのクラスに対応させて 5 つのオブジェクト指向プログラムのテンプレートに記憶させておく。また、各クラスのファイル名情報を各クラスに対応するテンプレートに付けておく。また、抽出／変換部 2200 は、各クラスにそれぞれ必要なメソッドを作り、中身は空にしておく。この時、意味情報を各メソッドに付与しておき、後で抽出変換の際に各メソッドを特定できるようにする。具体的には、ビュークラス 310 には初期化メソッドと画面表示入力メソッドを作成する。制御クラス 320 には初期化メソッドと UI メインメソッドを作成する。モデルクラス 330 には初期化メソッドと入力チェックメソッドと、画面データ受け取りメソッドを作成する。セッションクラス 340 には初期化メソッドとトランザクションチェックメソッドを作成す

る。エンティティクラス 3 5 0 には初期化メソッドとマスタファイル存在チェックメソッドとマッチング更新メソッドとを作成しておく。

【 0 0 6 6 】

次に、抽出／変換部 2 2 0 0 が中間プログラム 2 0 0 から必要な情報を抽出変換し、最終プログラム 3 0 0 を生成する動作について、まず、インターフェースクラス 2 1 0 からビュークラス 3 1 0、制御クラス 3 2 0 及びモデルクラス 3 3 0 を抽出、変換する動作を説明し、その後にファイルクラス 2 2 0 からセッションクラス 3 4 0 とエンティティクラス 3 5 0 とを抽出、変換する動作を説明する。

【 0 0 6 7 】

・ インターフェースクラス 2 1 0 から 3 つのクラスへの対応付け

まず、インターフェースクラス 2 1 0 から 3 つのクラスへの対応付けについて説明する。抽出／変換部 2 2 0 0 は、インターフェースクラス 2 1 0 の持つ役割のうち画面表示と入力の役割部分をビュークラス 3 1 0 に振り分ける。次に、抽出／変換部 2 2 0 0 は、入力のチェックの役割部分をモデルクラス 3 3 0 に振り分ける。そして、抽出／変換部 2 2 0 0 は、これら 2 つのクラスに振り分けた各役割を呼び出す動作制御を制御クラス 3 2 0 に振り分ける。

以上に説明したインターフェースクラス 2 1 0 から 3 つのクラスへの対応付けを図 3 7 に示す。また、各クラスへの具体的な抽出変換方法を図 3 8 ～図 4 0 に示す。ここで、図 3 8 ～図 4 0 中で示した位置番号は図 3 7 の位置番号に対応している。図 3 8 はインターフェースクラス 2 1 0 のプログラムからビュークラス 3 1 0 のプログラムへの具体的な抽出変換方法を示す。図 3 9 はインターフェースクラス 2 1 0 から制御クラス 3 2 0 への具体的な抽出変換方法を示す。図 4 0 はインターフェースクラス 2 1 0 からモデルクラス 3 3 0 への具体的な抽出変換方法を示す。図 3 8 に示す抽出、変換方法に基づいて抽出／変換部 2 2 0 0 により自動生成されたビュークラス 3 1 0 の具体的なプログラムのソースコードを図 4 1、図 4 2 に示す。なお、位置番号は変換前のプログラムのソースコードと変換後のプログラムのソースコードを対応付けるために記載されたものであり、実際のプログラム中には存在する必要はない。変換前のインターフェースクラス 2 1 0 は図

22、図23に明示されているが、このインターフェースクラス210のプログラム中に記載された位置番号(2-1)(2-2)(2-3)と図41、図42に示すビュークラス310のプログラム中に明示された位置番号(2-1)(2-2)(2-3)とが対応つけられている。

【0068】

図39に示す抽出、変換方法に基づいて抽出／変換部2200によって自動生成された制御クラス320のプログラムのソースコードを図43に示す。インターフェースクラス210と制御クラス320とは、これら2つのプログラムのソースコードに明示された位置番号(2-1)(2-4)によって対応付けがなされている。

【0069】

さらに、図40に示す抽出、変換方法に基づいて抽出／変換部2200によって自動生成されたモデルクラス330のプログラムのソースコードを図44、図45に示す。インターフェースクラス210とモデルクラス330とは、位置番号(2-1)(2-2)(2-5)(2-6)(2-7)(2-8)(2-9)によって対応付けがなされている。

【0070】

・ファイルクラス220から2つのクラスへの対応付け

次に、ファイルクラス220から2つのクラス(340, 350)への対応付けについて説明する。抽出／変換部2200は、ファイルクラス220の持つ役割をセッションクラス340及びエンティティクラス350の2つのクラスに次のように振り分ける。すなわち、ファイルクラス220では、マスタファイルの更新又は削除の場合には、トランザクションレコードに該当するものがマスタファイルに存在することを確認し、マスタファイルへの追加の場合には、トランザクションレコードに該当するものがマスタファイルに存在しないことを確認するロジックが存在していた。抽出／変換部2200は、これらのロジックをセッションクラス340に振り分ける。また、抽出／変換部2200は、トランザクションレコードの処理区分に応じてマスタファイルにトランザクションレコードを追加または更新または削除するロジックをエンティティクラス350に振り分け

る。

このような対応づけによりファイルクラス 2 2 0 のプログラムのソースコードは変換装置 B 2 0 0 0 によってセッションクラス 3 4 0 のプログラムのソースコードとエンティティクラス 3 5 0 のプログラムのソースコードに変換される。

【 0 0 7 1 】

上記抽出／変換部 2 2 0 0 によるファイルクラス 2 2 0 からセッションクラス 3 4 0 及びエンティティクラス 3 5 0 への対応付けを図 4 6 に示す。

また、ファイルクラス 2 2 0 のプログラムからセッションクラス 3 4 0 のプログラムへ変更するための具体的抽出変換方法を図 4 7 に示す。また、ファイルクラス 2 2 0 のプログラムからエンティティクラス 3 5 0 のプログラムへ変換するための具体的抽出変換方法を図 4 8 に示す。

抽出／変換部 2 2 0 0 は、図 4 7 に示す抽出変換方法に基づいて、ファイルクラス 2 2 0 のプログラムから必要なデータを抽出し、抽出したデータを用いてセッションクラス 3 4 0 のプログラムに自動変換する。自動変換されたセッションクラス 3 4 0 のプログラムを図 4 9、図 5 0 に示す。

また、抽出／変換部 2 2 0 0 は、図 4 8 に示す抽出変換方法に基づいて、ファイルクラス 2 2 0 のプログラムから必要なデータを抽出し、抽出したデータを用いてエンティティクラス 3 5 0 のプログラムに自動変換する。自動変換されたエンティティクラス 3 5 0 のプログラムを図 5 1～図 5 3 に示す。

【 0 0 7 2 】

上記生成された最終プログラム 3 0 0 の 5 つのクラスには、中間プログラム 2 0 0 には含まれない要素が存在する。そのため、その要素を新規に生成する必要がある。新規に作成すべき各クラスへの追加の要素を説明する。

この追加の要素作成は、抽出／変換部 2 2 0 0 によって行なわれる。

【 0 0 7 3 】

まず、抽出／変換部 2 2 0 0 がビュークラス 3 1 0 への新規事項追加を行う動作について図 4 1 を用いて説明する。

・初期化メソッド

初期化メソッドの内容として自己インスタンスを生成し、制御クラスの初期化

メソッドを呼び出すステップを追加する。

具体的には、メソッドのデータ部にWORKING-STORAGE SECTIONを設け、自己インスタンスを参照するステップを追加する（ステップ19～20）。メソッドの手続き部には、自己インスタンスを生成するステップ（ステップ22～ステップ23）、また自己インスタンスを引数として制御クラスの初期化メソッドを呼び出すステップ（ステップ24～ステップ25）を追加する。

【0074】

次に、抽出／変換部2200が制御クラス320へ追加する事項について図43を用いて説明する。

・初期化メソッド

初期化メソッドの内容として自己インスタンスを生成し、引数のビューインスタンスと自己インスタンスとをつなげるステップを追加する。またモデルクラス330の初期化メソッドを呼び出し、戻り値のモデルインスタンスを自己インスタンスとつなげるステップを追加する。

具体的には、メソッドのデータ部にLINKAGE SECTIONを設け、引数のビューインスタンスを参照するステップを追加する（ステップ20～ステップ21）。またWORKING-STORAGE SECTIONを設け、自己インスタンスを参照するステップを追加する（ステップ22～ステップ23）。メソッドの手続き部には、ビューインスタンスを引数として受け取ることを宣言するステップ（ステップ24）、自己インスタンスを生成するステップ（ステップ25～ステップ26）、引数のビューインスタンスと自己インスタンスとをつなげるステップ（ステップ27～ステップ28）、モデルクラス330の初期化メソッドを呼び出し、戻り値のモデルインスタンスを自己インスタンスとつなげるステップ（ステップ29～ステップ31）、また制御クラス320のUIメインメソッドを呼び出すステップ（ステップ32）を追加する。

【0075】

次に、抽出／変換部2200がモデルクラス330へ追加する事項について図44、図45を用いて説明する。

・ 初期化メソッド

初期化メソッドの内容として自己インスタンスを生成し、引数のビューインスタンスと自己インスタンスとをつなげ、自己インスタンスを戻り値として設定するステップを追加する。

具体的には、メソッドのデータ部にLINKAGE SECTIONを設け、引数のビューインスタンスを参照するステップと戻り値の自己インスタンスを参照するステップを追加する（ステップ20～ステップ22）。またWORKING-STORAGE SECTIONを設け、自己インスタンスを参照するステップを追加する（ステップ23～ステップ24）。メソッドの手続き部には、ビューインスタンスを引数として受取り自己インスタンスを戻り値として返すことを宣言するステップ（ステップ26～ステップ27）、自己インスタンスを生成するステップ（ステップ28～ステップ29）、引数のビューインスタンスと自己インスタンスとをつなげるステップ（ステップ30～ステップ31）、また自己インスタンスを戻り値として設定するステップ（ステップ32）を追加する。

【0076】

次に、抽出／変換部2200がセッションクラス340へ追加する事項について図49、図50を用いて説明する。

・ 初期化メソッド

初期化メソッドの内容として自己インスタンスを生成し、エンティティクラス350の初期化メソッドを呼び出して戻り値のエンティティインスタンスを自己インスタンスにつなげ、自己インスタンスを戻り値として設定するステップを追加する。

具体的には、メソッドのデータ部にLINKAGE SECTIONを設け、戻り値の自己インスタンスを参照するステップを追加する（ステップ18～ステップ19）。またWORKING-STORAGE SECTIONを設け、自己インスタンスを参照するステップを追加する（ステップ20～ステップ21）。メソッドの手続き部には、自己インスタンスを戻り値として返すことを宣言するステップ（ステップ22）、自己インスタンスを生成するステップ（ステップ23～ステップ24）、エンティティクラスの初期化メソッドを呼び出し、戻り

値のエンティティインスタンスを自己インスタンスとつなげるステップ（ステップ25～ステップ26）、また自己インスタンスを戻り値として設定するステップ（ステップ27）を追加する。

・トランザクションチェックメソッド

トランザクションチェックメソッドにはエンティティクラス350のマスタレコード存在チェックメソッドを呼び出し、結果が正しければエンティティクラス350のマッチング更新メソッドを呼び出すステップを追加する。

具体的には、メソッドのデータ部にWORKING-STORAGE SECTIONを設け、エンティティクラス350のマスタレコード存在チェックメソッドからの戻り値を格納するステップを追加する（ステップ61～ステップ64）。メソッドの手続き部には、エンティティクラス350のマスタレコード存在チェックメソッドを呼び出すステップ（ステップ67～ステップ68）、またこの呼び出しの結果が正しければエンティティクラス350のマッチング更新メソッドを呼び出すステップ（ステップ69～ステップ82）を追加する。

【0077】

最後に抽出／変換部2200がエンティティクラス350のプログラムへ追加する事項について図51、図52を用いて説明する。

・初期化メソッド

初期化メソッドの内容として、自己インスタンスを生成し、これを戻り値として返すステップを追加する。

具体的にはメソッドのデータ部にLINKAGE SECTIONを設け、戻り値の自己インスタンスを参照するステップを追加する（ステップ17～ステップ18）。またWORKING-STORAGE SECTIONを設け、自己インスタンスを参照するステップを追加する（ステップ19～ステップ20）。メソッドの手続き部には、戻り値として自己インスタンスを返すことを宣言するステップ（ステップ21）、自己インスタンスを生成するステップ（ステップ22～ステップ23）、戻り値に自己インスタンスを設定するステップ（ステップ24）を追加する。

・マスタレコード存在チェックメソッド

マスタレコード存在チェックメソッドの内容として、レコードキーを引数に受取り、そのキーでマスタファイルを読み込み、その結果をマスタレコード存在フラグに格納して戻り値として返すステップを追加する。

具体的には、メソッドのデータ部にLINKAGE SECTIONを設け、引数のレコードキーを格納するステップと戻り値のマスタレコード存在フラグを格納するステップを追加する（ステップ62～ステップ66）。メソッドの手続き部には、レコードキーを引数として受取りマスタレコード存在フラグを戻り値として返すことを宣言するステップ（ステップ67～ステップ68）、マスタファイルを読み込む準備をするステップ（ステップ70～ステップ71）、またマスタファイルを読み込みその結果を戻り値として設定するステップ（ステップ72～ステップ77）を追加する。

【0078】

このようにして変換装置B2000は中間プログラム200のソースコードから最終プログラム300のソースコードを自動変換することにより、よりオブジェクト指向性の高いプログラムを生成することができる。このようにオブジェクト指向性の高いプログラムに変換することによって、オブジェクト内部の仕様変更が外部に及ばないようなプログラムが可能になり、ソースコードを再利用しやすくなる。よって、過去において作成されたプログラム資産を分散型処理でさらに有効活用することが可能となる。

【0079】

また、WEBやXMLなどに容易に連携することができるため、中間プログラム200よりも現代社会のネットワークシステムにより合致したプログラム構造と言える。従って、このようにして、従来のプログラムから自動的に人的労力をかけずに自動変換された最終プログラム300によれば、現在主流となっている分散型処理のインフラをより有効に活用することができるアプリケーションプログラムとして再利用することが可能となる。

【0080】

実施の形態2.

次に実施の形態2について説明する。本実施の形態では、実施の形態1のよう

に中間プログラム 200 を生成するステップを設けず、コボルプログラム 100 のソースコードから直接最終プログラム 300 のソースコードを抽出、変換する形態である。

図 5 4 は、本実施の形態の概念図である。

本実施の形態では、変換装置 C 3000 がコボルプログラム 100 のソースコードを直接最終プログラム 300 のソースコードに変換している。このように、変換装置 C 3000 が自動的に直接最終プログラム変換を行うことにより、中間プログラム 200 を生成する段階を設けずに、集中型処理から WEB や XML などに連携できる分散型処理に適したプログラムを短期間に容易に取得することができる。

【0081】

変換装置 C 3000 の構成及び動作について説明する。図 5 5 は、変換装置 C 3000 の内部構成図である。

実施の形態 1 の変換装置 A 1000 の内部構成を示す図 4 と比べ、内部構成自体は同一である。ただし、出力部 1700 が出力するプログラムが最終プログラム 300 であることと、抽出／変換部 3100 の動作が一部異なっている。即ち、変換装置 A 1000 の抽出／変換部 1600 では中間プログラム 200 を生成するためにデータの抽出変換を行っていたが、本実施の形態の抽出／変換部 3100 では、最終プログラム 300 を生成するためにデータの抽出及び変換を行なっている。

【0082】

入力部 1100 はコボルプログラム 100 を入力し、記憶部 1800 に記憶する。分割部 1200 はコボルプログラム 100 をまとまりある複数のプログラムに分解し、構文解析部 1300 は分解されたそれぞれのプログラムの構文を解析する。プログラム判断部 1400 は各プログラムの役割を判断し、節判断部 1500 は各プログラム中の節の内容、役割を判断する。

これらの動作を終えた後、抽出／変換部 3100 は最終プログラム 300 を生成するためのデータの抽出及び変換を行ない、その結果、生成された最終プログラム 300 は出力部 1700 によって出力される。この場合、記憶部 1800 は

必要に応じてデータを記憶する領域として利用することができ、また 3 0 0 0 の内部に記憶部 1 8 0 0 が存在しなくても、外部記憶装置に記憶させてもよい。

【 0 0 8 3 】

このように、本実施の形態では、コボルプログラム 1 0 0 から中間プログラム 2 0 0 を出力することなく、直接最終プログラム 3 0 0 を出力するため、変換処理を高速に行え、WEB や XML などに連携できる分散型処理に適したプログラムを短期間に容易に取得することができる。

【 0 0 8 4 】

また、最終プログラム 3 0 0 を中間プログラム 2 0 0 と比較すると機能自体は変わらないが、最終プログラム 3 0 0 ではプログラムの部品化が進むため、既存部品に差分だけを付け加えて必要な動作を実行するプログラムを容易に作成できる。このため、さらに資産価値が高いプログラムを取得することができる。

【 0 0 8 5 】

また、最終プログラム 3 0 0 を中間プログラム 2 0 0 を J a v a（登録商標）言語や C ++ 言語等の言語を用いて記述することが可能である。

【 0 0 8 6 】

なお、上記全ての実施の形態では、コボルプログラムを基に変換プログラムを生成したが、変換前のプログラムはコボルプログラムに限る必要はなく、バッチ処理をする構造化されたプログラムであればよい。従って、構造化されたプログラムであれば、上記変換装置により、分散型処理に適合したプログラムに変換することができる。

【 0 0 8 7 】

図 5 6 は、変換装置 A、変換装置 B、変換装置 C のコンピュータ基本構成図である。

図 5 6 において、プログラムを実行する CPU（C e n t r a l P r o c e s s i n g U n i t）4 0 は、バス 3 8 を介してモニタ 4 1、キーボード 4 2、マウス 4 3、通信ボード 4 4、磁気ディスク装置 4 6 等と接続されている。

磁気ディスク装置 4 6 には、オペレーティングシステム（OS）4 7、プログラム群 4 9、ファイル群 5 0 が記憶されている。ただし、プログラム群 4 9、フ

ファイル群 5 0 が一体となってオブジェクト指向のプログラム群 4 9 を形成する形態も一実施の形態として考えられる。

プログラム群 4 9 は、CPU 4 0、OS 4 7 により実行される。

上記各実施の形態では、変換装置 A、変換装置 B、変換装置 C は、通信ボード 4 4 の機能を使用して、各種ネットワークを経由して接続された機器と通信を行う。

【0 0 8 8】

以上に記載した「格納する」、「記憶する」という用語は、記録媒体に保存することを意味する。

【0 0 8 9】

すべての実施の形態では、各構成要素の各動作はお互いに関連しており、各構成要素の動作は、上記に示された動作の関連を考慮しながら、一連の動作として置き換えることができる。そして、このように置き換えることにより、変換装置の実施形態を変換方法の発明の実施形態とすることができる。

また、上記各構成要素の動作を、各構成要素の処理と置き換えることにより、変換プログラムの実施の形態とすることができる。

また、変換プログラムをコンピュータ読み取り可能な記録媒体に記憶させることで、変換プログラムを記録したコンピュータ読み取り可能な記録媒体の実施の形態とすることができる。

【0 0 9 0】

変換プログラムの実施の形態及び変換プログラムを記録したコンピュータ読み取り可能な記録媒体の実施の形態は、すべてコンピュータで動作可能なプログラムにより構成することができる。

また、変換プログラムの実施の形態および変換プログラムを記録したコンピュータ読み取り可能な記録媒体の実施の形態における各処理はプログラムで実行されるが、このプログラムは、記録装置に記録されていて、記録装置から中央処理装置（CPU）に読み込まれ、中央処理装置によって、プログラムに記述された動作が実行されることになる。

【0 0 9 1】

また、各実施の形態に記載されたソフトウェアやプログラムは、ROM (READ ONLY MEMORY) に記憶されたファームウェアで実現されていても構わない。あるいは、ソフトウェアとファームウェアとハードウェアとの組み合わせで前述したプログラムの各機能を実現しても構わない。

【0092】

【発明の効果】

本発明の実施の形態によれば、旧来のプログラム資産を分散システムに対応するプログラム資産に変換することができる。

【0093】

また、本発明の実施の形態によれば、旧来コーディングされたプログラム中の業務ロジックを再利用することができる。

【0094】

また、本発明の実施の形態によれば、旧来のプログラム資産を利用して、新しいシステムを構築することができる。

【0095】

また、本発明の実施の形態によれば、旧来のプログラム資産から中間プログラムへ変換することなく直接最終プログラムへ変換することができる。

【図面の簡単な説明】

【図1】 プログラム変換方法の一例を示す図である。

【図2】 オフライン一括処理をオンライン一件別処理に変換する図である。

【図3】 オフライン一括処理をオンライン一件別処理に変換する図である。

【図4】 変換装置A1000の内部構成図である。

【図5】 コボルプログラム100から中間プログラム200を取得する動作を示した図である。

【図6】 コボルプログラム100の一例を示す図である。

【図7】 コボルプログラム100を分割する流れ図である。

【図8】 各プログラムの役割を判断する流れ図である。

- 【図 9】 入力チェックプログラム 1 0 2 の節の構造を示す図である。
- 【図 1 0】 各節の役割を判断する流れ図である。
- 【図 1 1】 マッチングプログラム 1 0 7 の節の構造を示す図である。
- 【図 1 2】 各節の役割を判断する流れ図である。
- 【図 1 3】 入力チェックプログラム 1 0 2 と結果出力プログラム 1 0 9 とインターフェースクラス 2 1 0 の対応図である。
- 【図 1 4】 マッチングプログラム 1 0 7 とソートプログラム 1 0 4 とファイルクラス 2 2 0 の対応図である。
- 【図 1 5】 プログラムの抽出、変換の詳細を示す図である。
- 【図 1 6】 プログラムの抽出、変換の詳細を示す図である。
- 【図 1 7】 入力チェックプログラム 1 0 2 の一部を示す図である。
- 【図 1 8】 インターフェースクラス 2 1 0 の一部を示す図である。
- 【図 1 9】 入力チェックプログラム 1 0 2 の一部を示す図である。
- 【図 2 0】 インターフェースクラス 2 1 0 の一部を示す図である。
- 【図 2 1】 入力チェックプログラム 1 0 2 を示す図である。
- 【図 2 2】 インターフェースクラス 2 1 0 を示す図である。
- 【図 2 3】 インターフェースクラス 2 1 0 を示す図である。
- 【図 2 4】 結果出力プログラム 1 0 9 を示す図である。
- 【図 2 5】 結果出力プログラム 1 0 9 を示す図である。
- 【図 2 6】 プログラムの抽出、変換の詳細を示す図である。
- 【図 2 7】 プログラムの抽出、変換の詳細を示す図である。
- 【図 2 8】 プログラムの抽出、変換の詳細を示す図である。
- 【図 2 9】 マッチングプログラム 1 0 7 を示す図である。
- 【図 3 0】 マッチングプログラム 1 0 7 を示す図である。
- 【図 3 1】 マッチングプログラム 1 0 7 を示す図である。
- 【図 3 2】 ソートプログラム 1 0 4 を示す図である。
- 【図 3 3】 ファイルクラス 2 2 0 のプログラムを示す図である。
- 【図 3 4】 ファイルクラス 2 2 0 のプログラムを示す図である。
- 【図 3 5】 変換装置 B 2 0 0 0 の内部構成図である。

【図 3 6】 オンライン一件別処理をよりオブジェクト指向的なオンライン一件別処理に変換する図である。

【図 3 7】 インターフェースクラス 2 1 0 と 3 つのクラスの対応図である。

【図 3 8】 プログラムの抽出、変換の詳細を示す図である。

【図 3 9】 プログラムの抽出、変換の詳細を示す図である。

【図 4 0】 プログラムの抽出、変換の詳細を示す図である。

【図 4 1】 ビュークラス 3 1 0 のプログラムを示す図である。

【図 4 2】 ビュークラス 3 1 0 のプログラムを示す図である。

【図 4 3】 制御クラス 3 2 0 のプログラムを示す図である。

【図 4 4】 モデルクラス 3 3 0 のプログラムを示す図である。

【図 4 5】 モデルクラス 3 3 0 のプログラムを示す図である。

【図 4 6】 ファイルクラス 2 2 0 と 2 つのクラスの対応図である。

【図 4 7】 プログラムの抽出、変換の詳細を示す図である。

【図 4 8】 プログラムの抽出、変換の詳細を示す図である。

【図 4 9】 セッションクラス 3 4 0 のプログラムを示す図である。

【図 5 0】 セッションクラス 3 4 0 のプログラムを示す図である。

【図 5 1】 エンティティクラス 3 5 0 のプログラムを示す図である。

【図 5 2】 エンティティクラス 3 5 0 のプログラムを示す図である。

【図 5 3】 エンティティクラス 3 5 0 のプログラムを示す図である。

【図 5 4】 プログラム変換方法の他の一例を示す図である。

【図 5 5】 変換装置 C 3 0 0 0 の内部構成図である。

【図 5 6】 変換装置 A、変換装置 B、変換装置 C のコンピュータ基本構成図である。

【図 5 7】 従来図である。

【符号の説明】

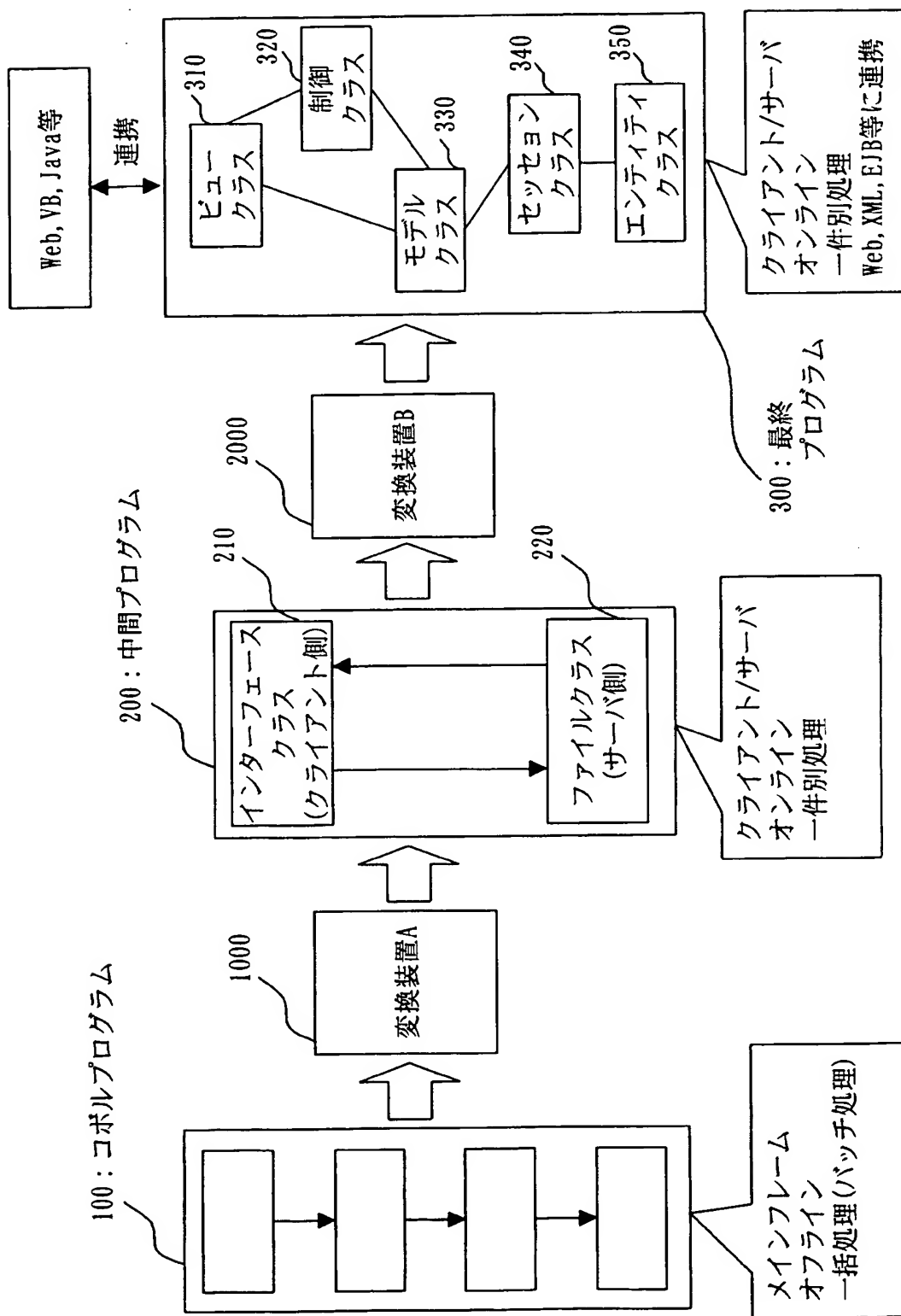
1 0 0 コボルプログラム、1 0 1 トランザクションファイル、1 0 2 入力チェックプログラム、1 0 3 チェック済みトランザクションファイル、1 0 4 ソートプログラム、1 0 5 チェック・ソート済みトランザクションファイル

ル、106 旧マスタファイル、107 マッチングプログラム、108 新マスタファイル、109 結果出力プログラム、200 中間プログラム、206 マスタファイル、210 インターフェースクラス、220 ファイルクラス、300 最終プログラム、310 ビュークラス、320 制御クラス、330 モデルクラス、340 セッションクラス、350 エンティティクラス、1000 変換装置A、1100 入力部、1200 分割部、1300 構文解析部、1400 プログラム判断部、1500 節判断部、1600 抽出／変換部、1700 出力部、1800 記憶部、2000 変換装置B、2100 入力部、2200 抽出／変換部、2300 出力部、2400 記憶部、3000 変換装置C、3100 抽出／変換部、4000 メインフレーム機器、5000 クライアント機器、6000 サーバ機器。

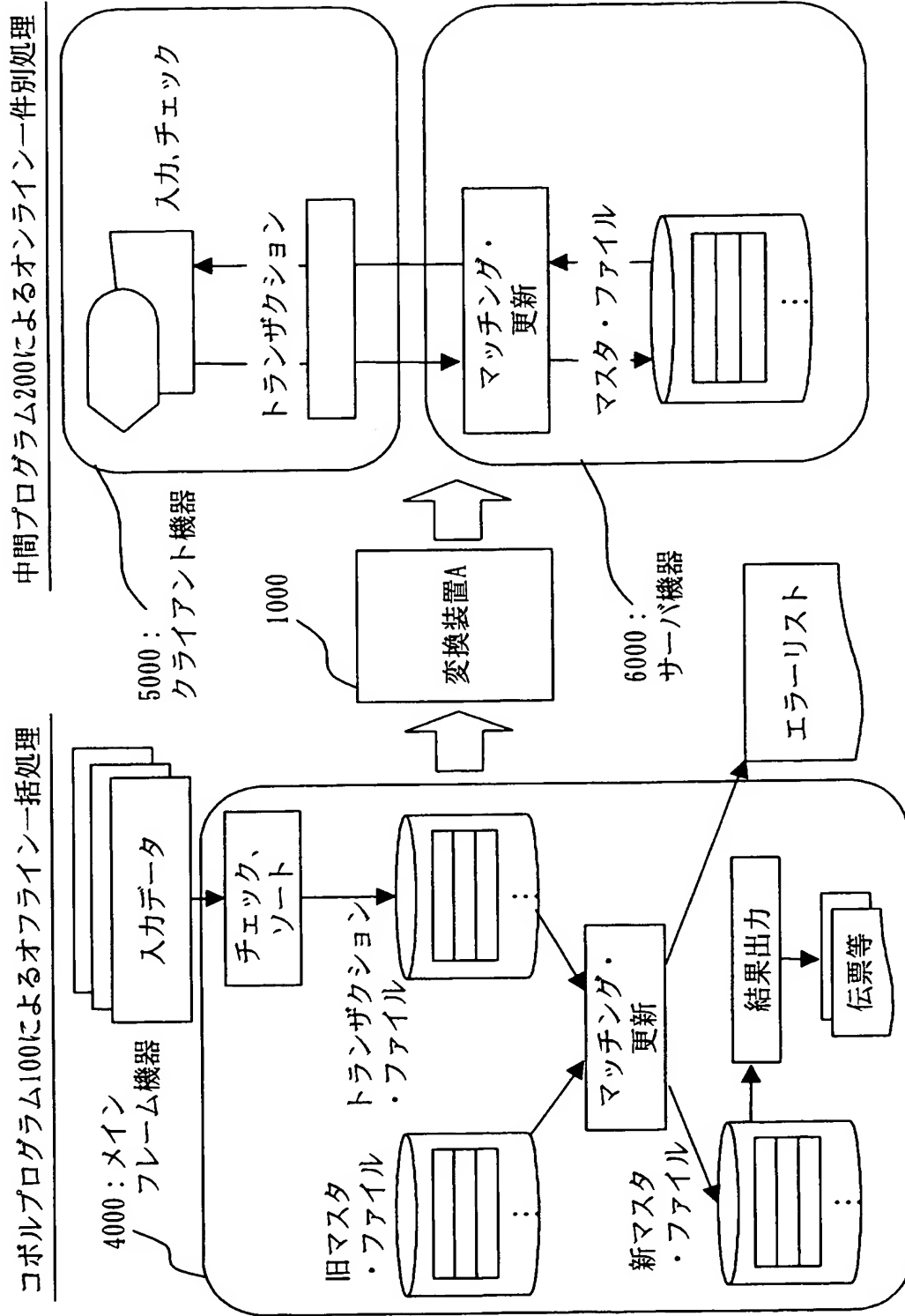
【書類名】

図面

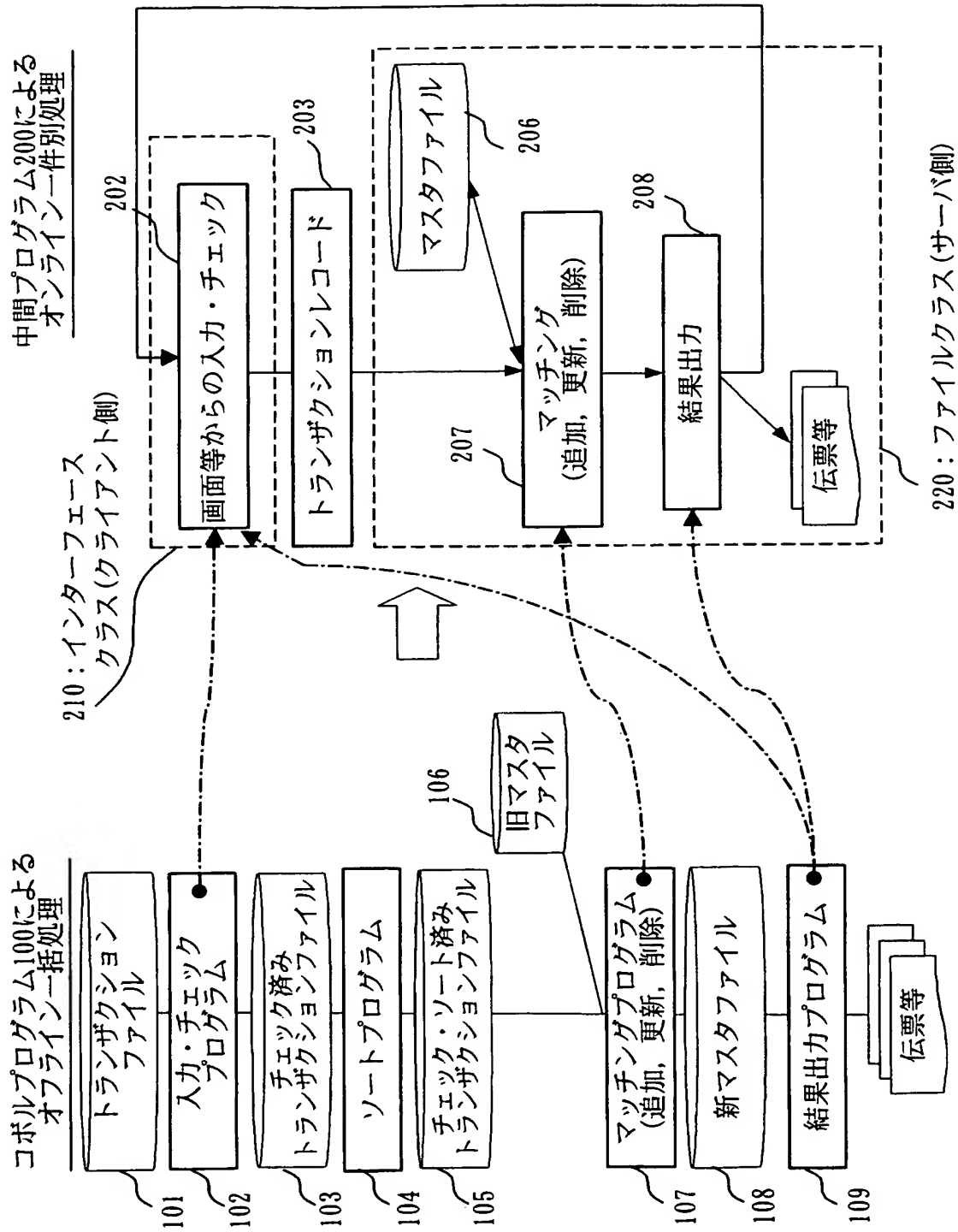
【図 1】



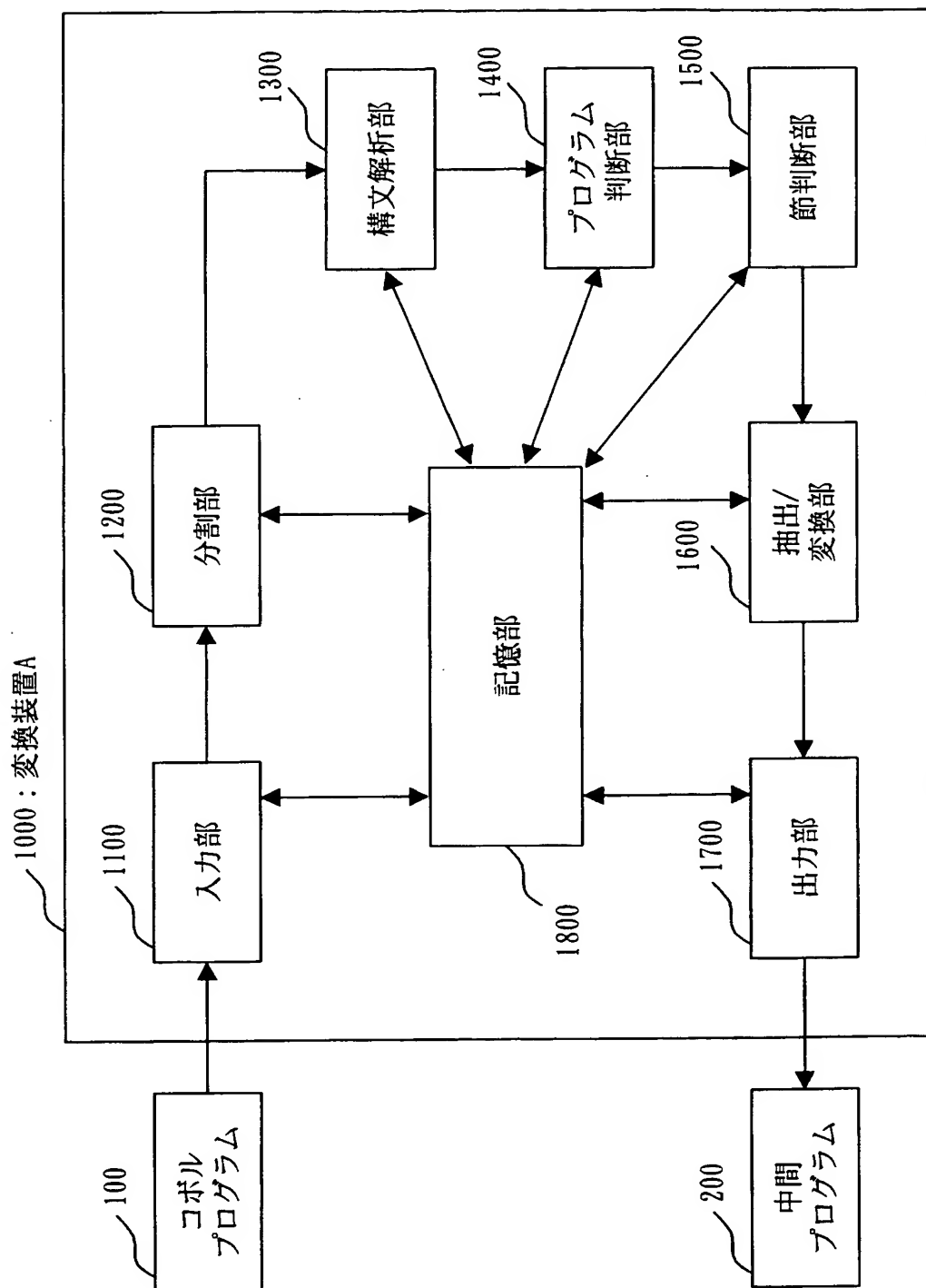
【図 2】



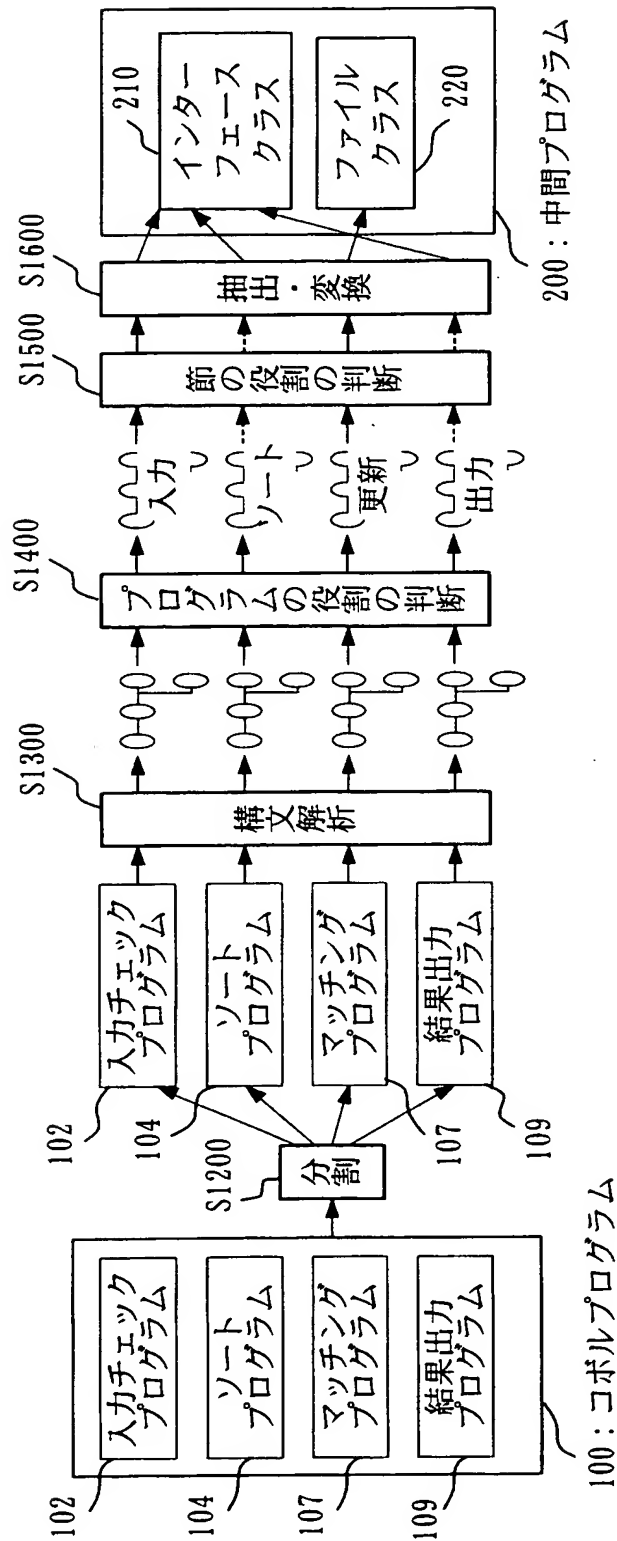
【図 3】



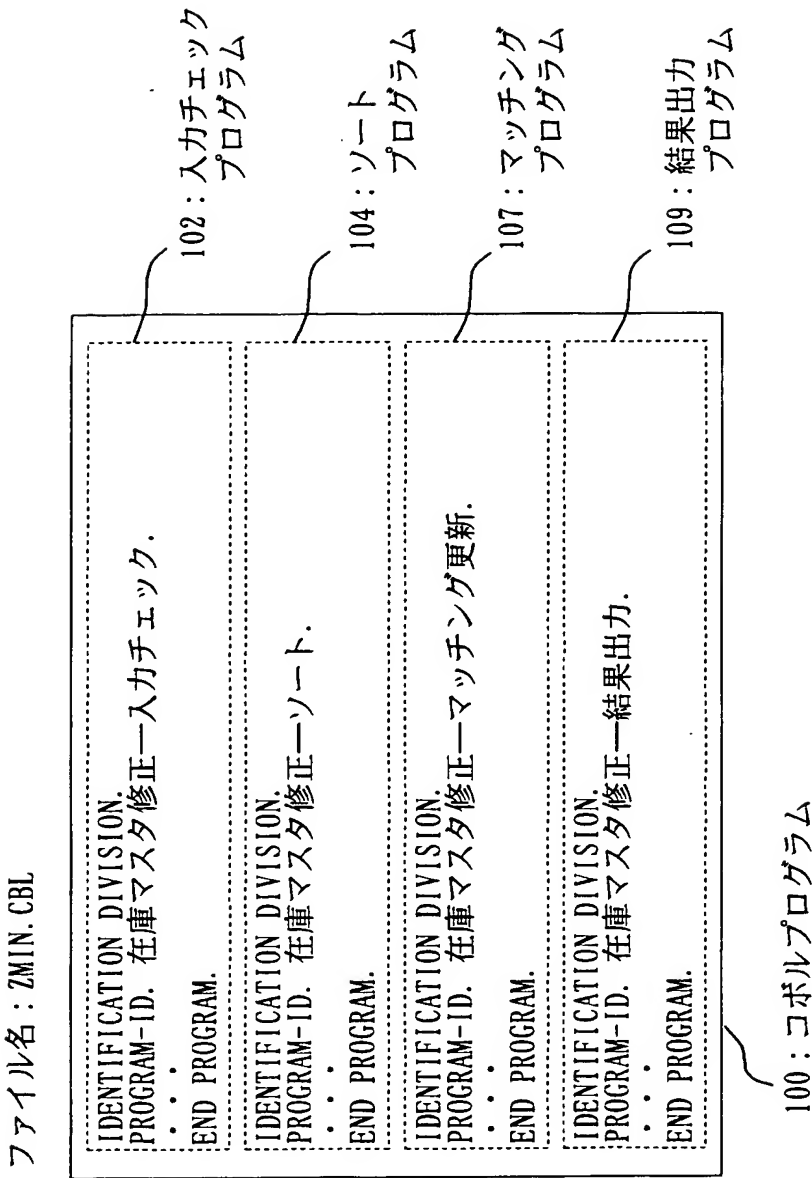
【図 4】



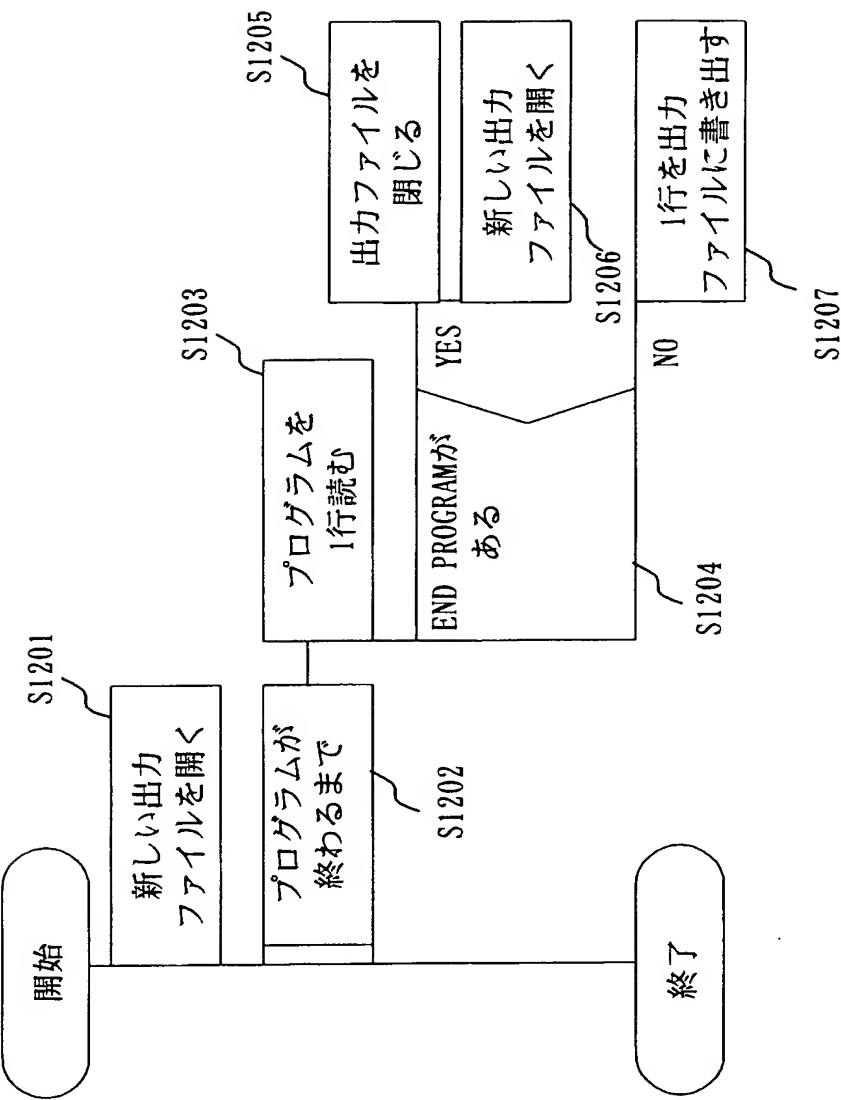
【図 5】



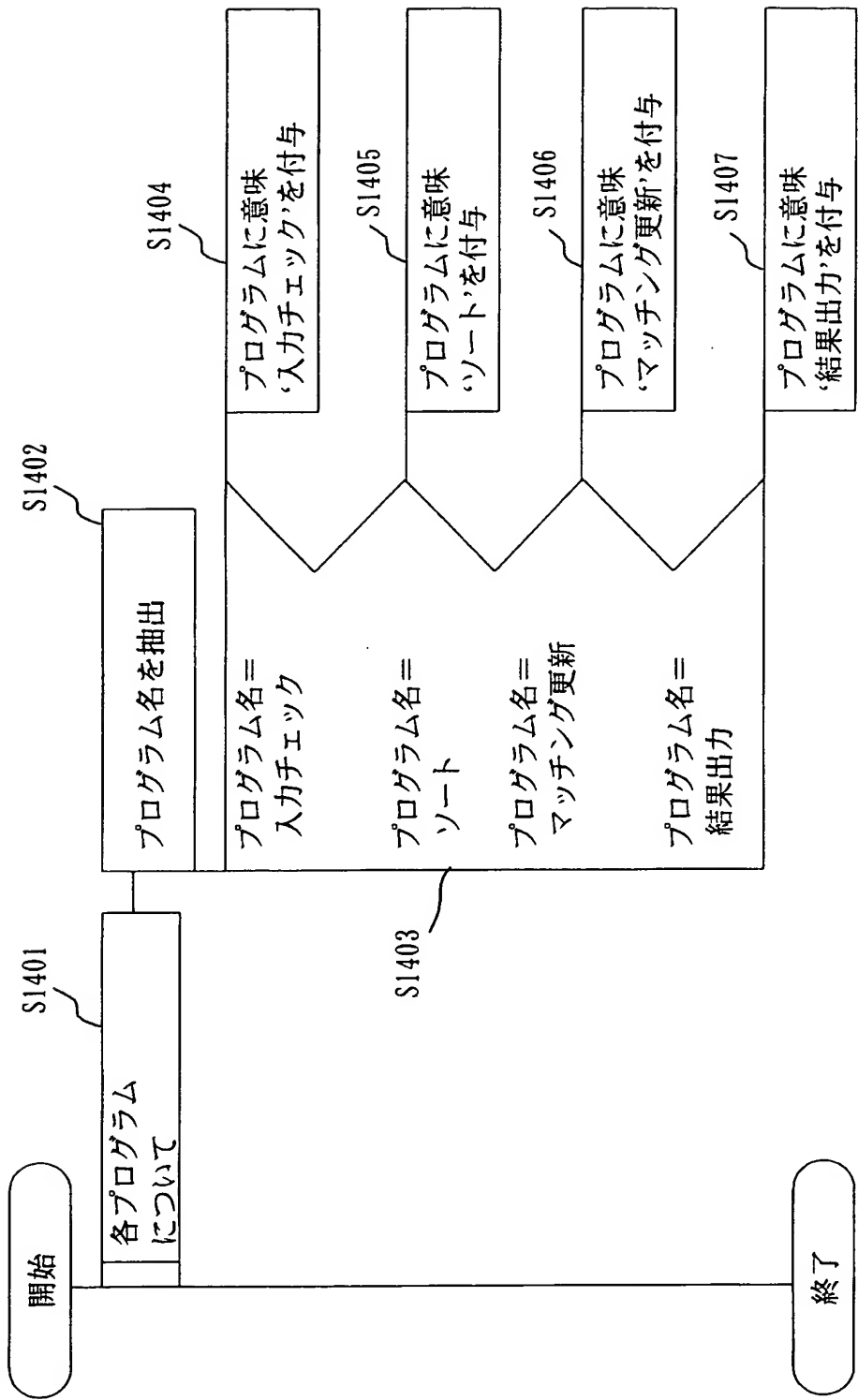
【図 6】



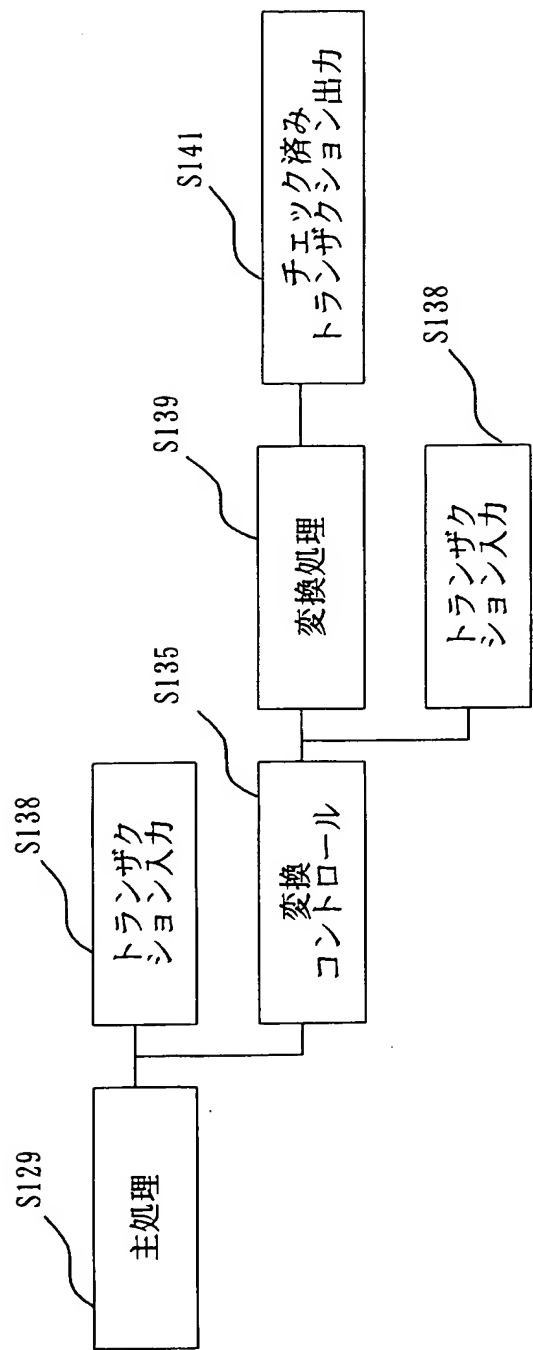
【図 7】



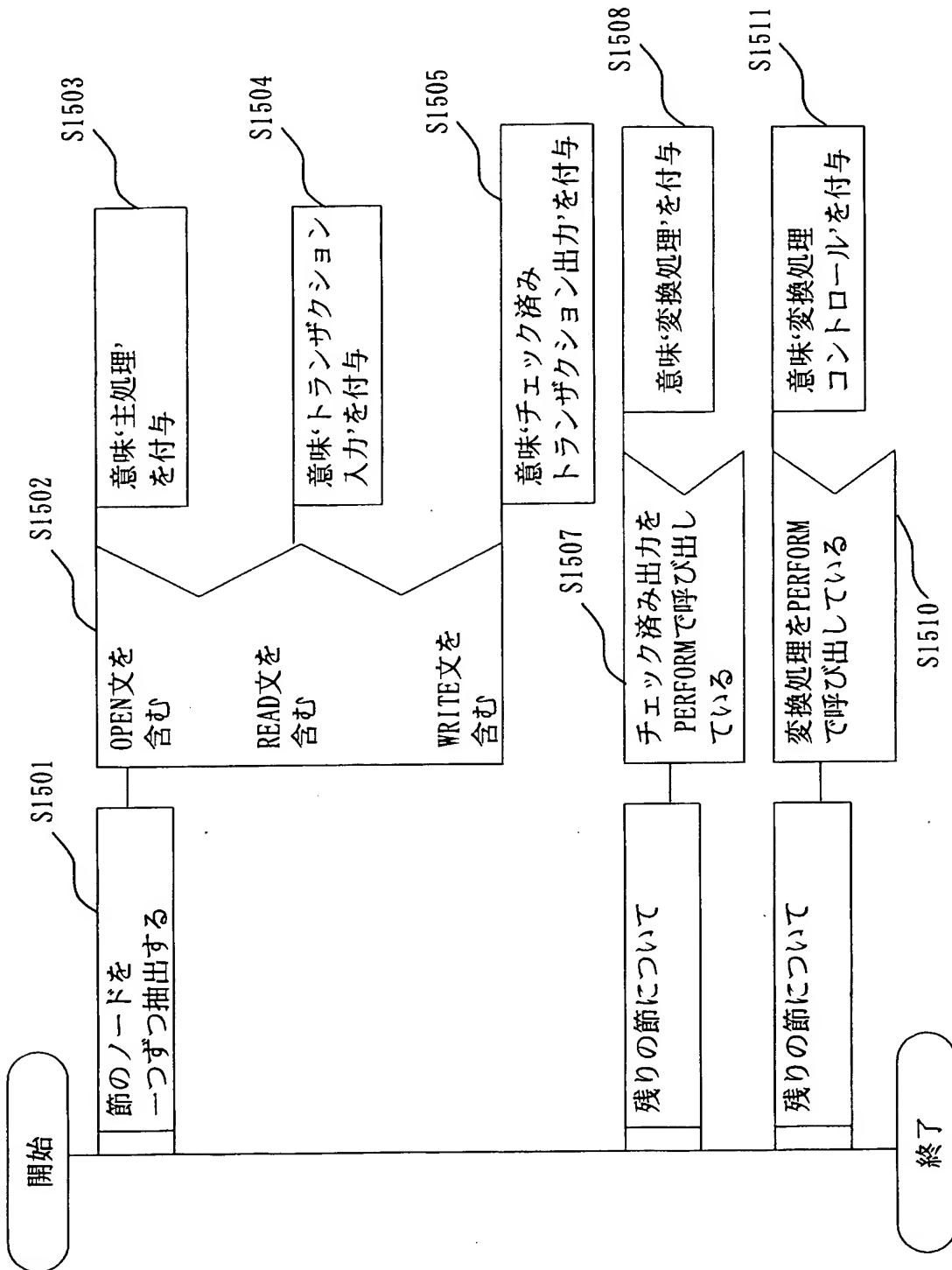
【図 8】



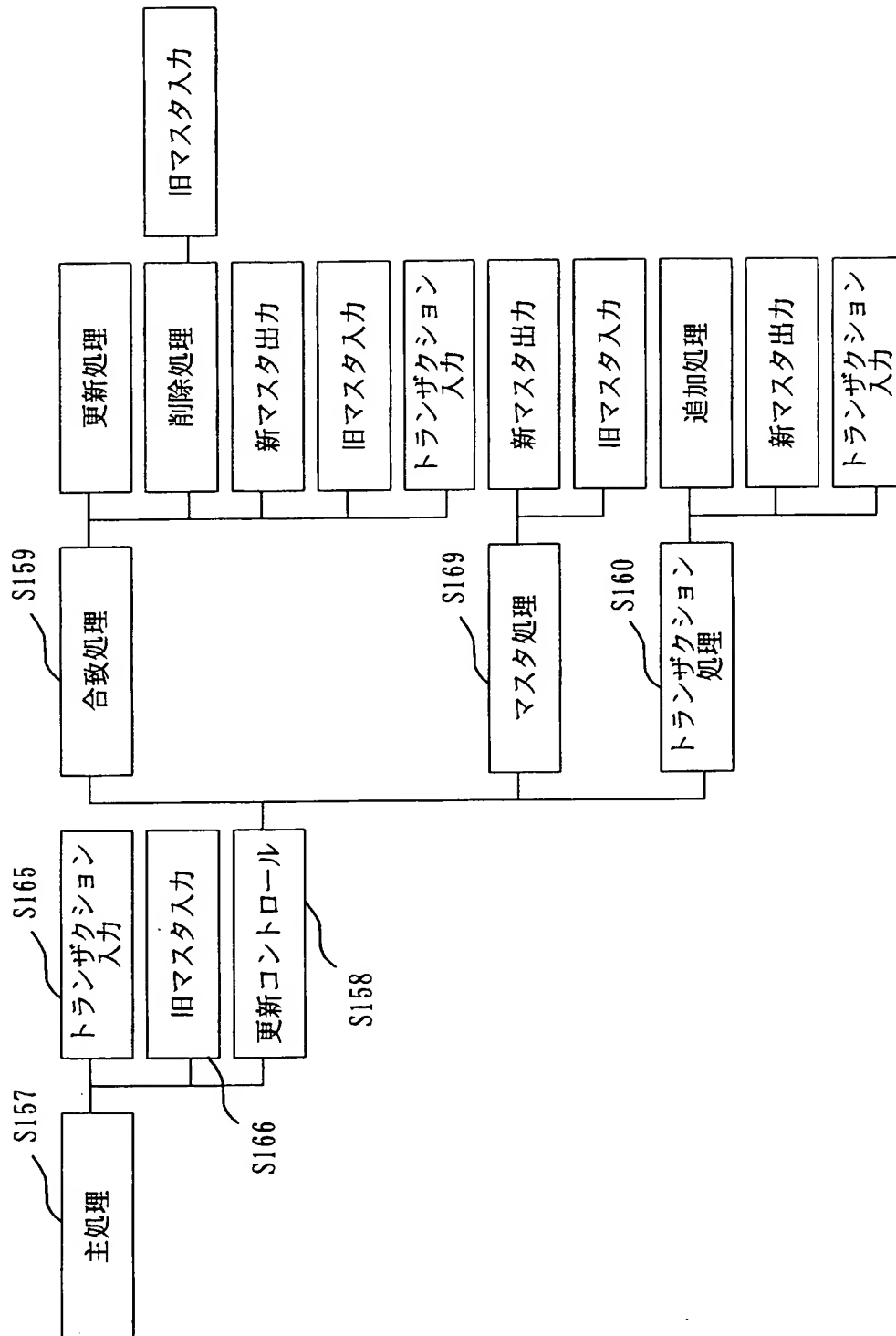
【図9】



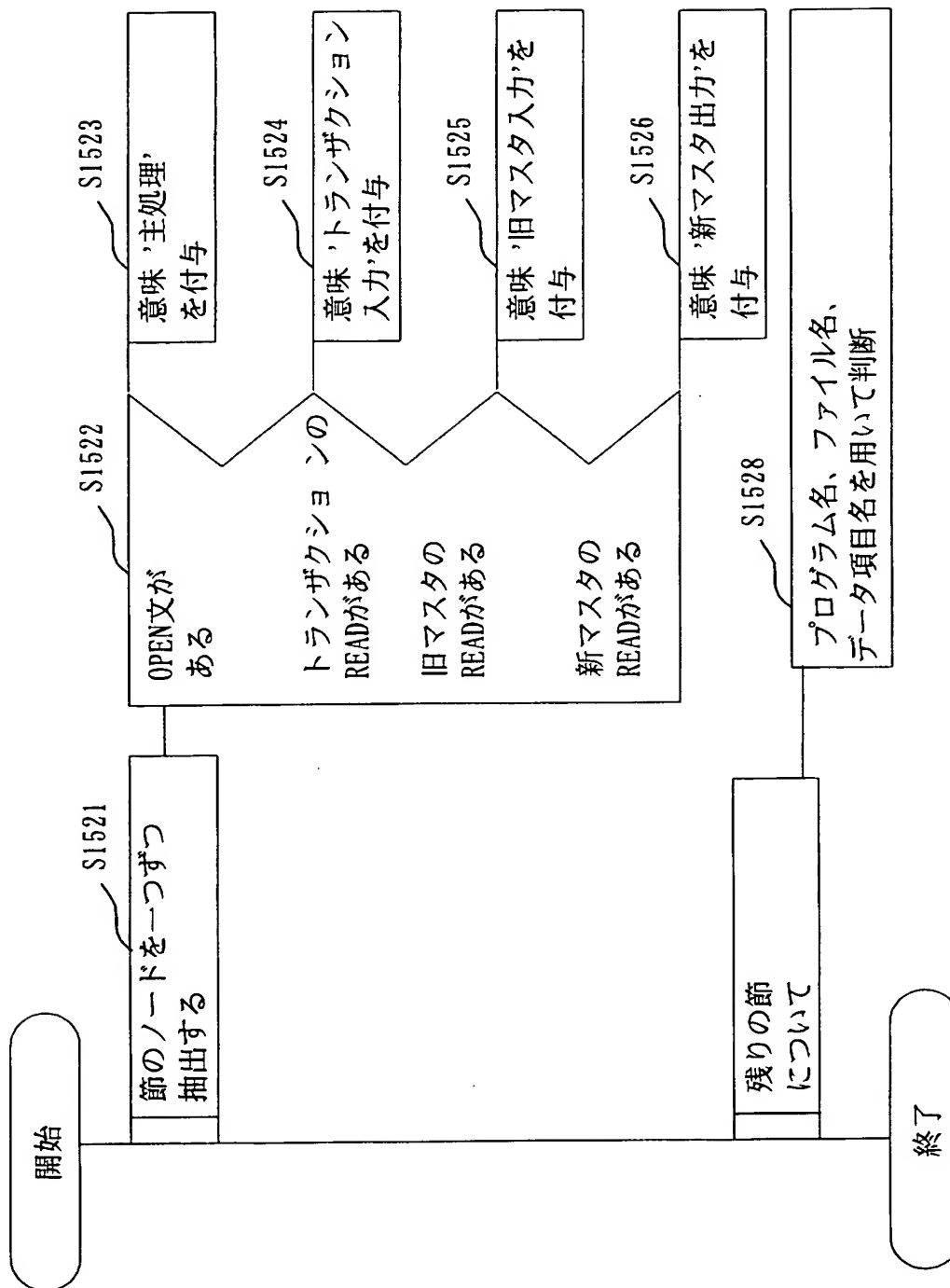
【図 10】



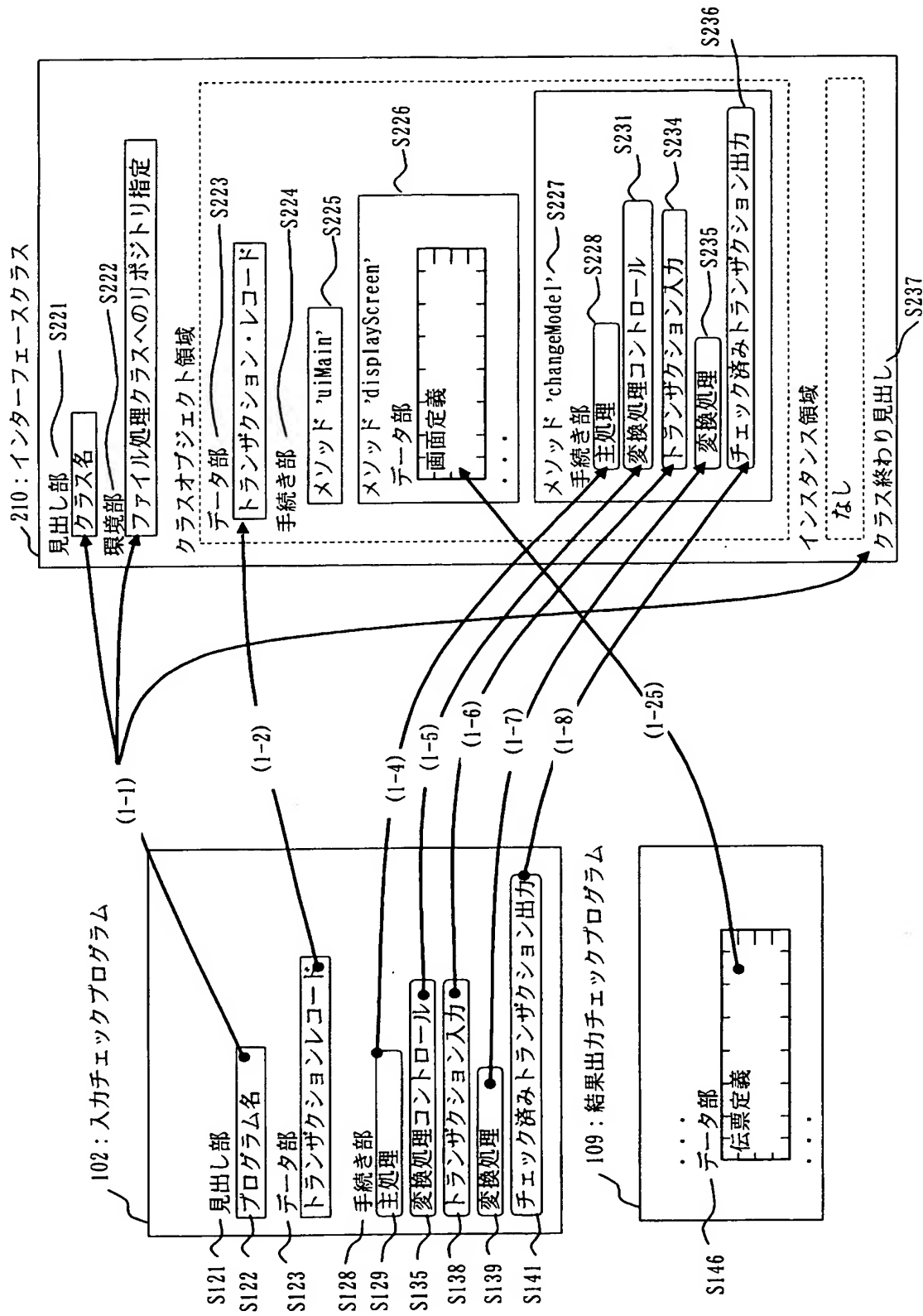
【図 11】



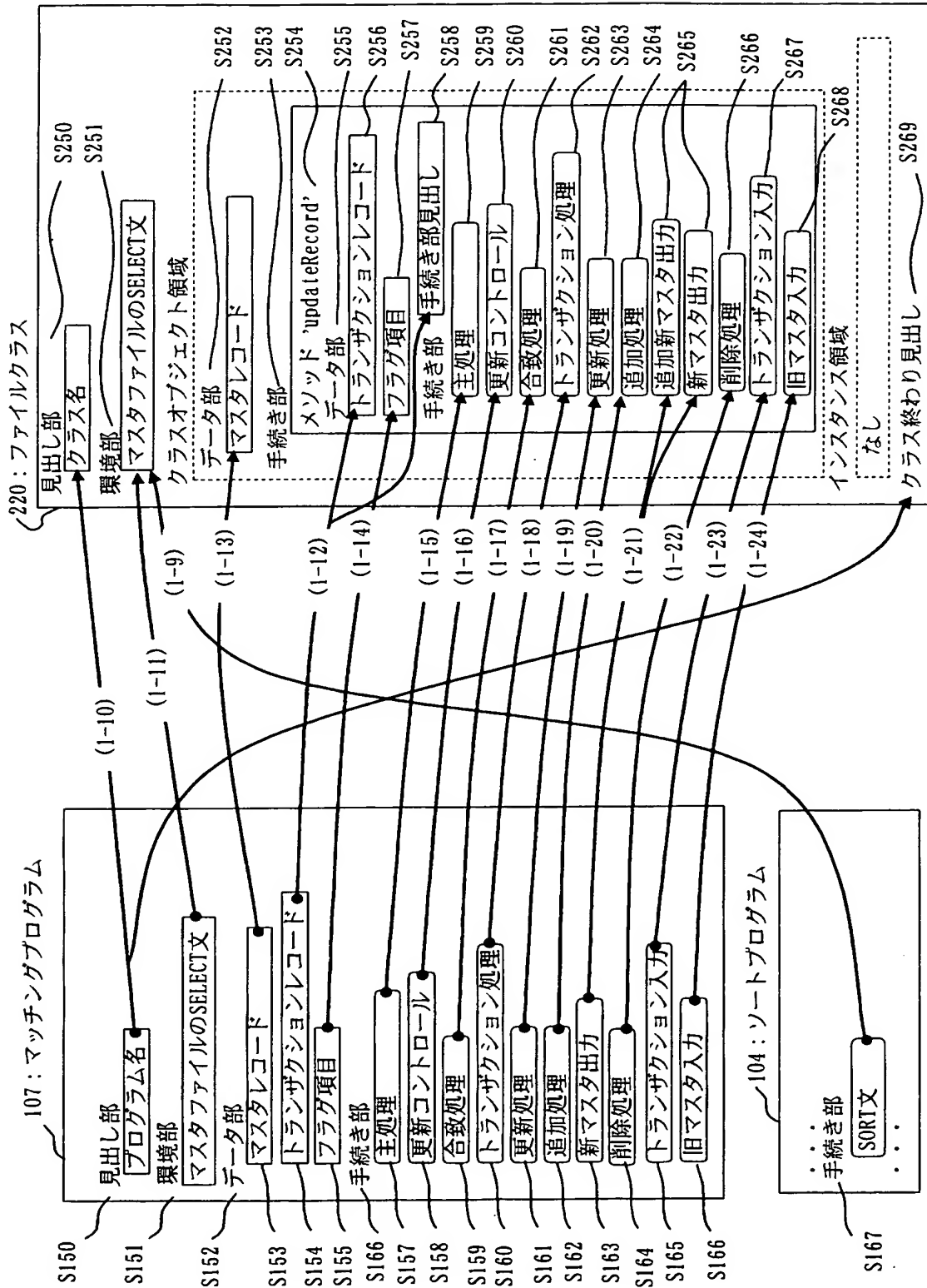
【図 12】



【図 13】



【図 14】



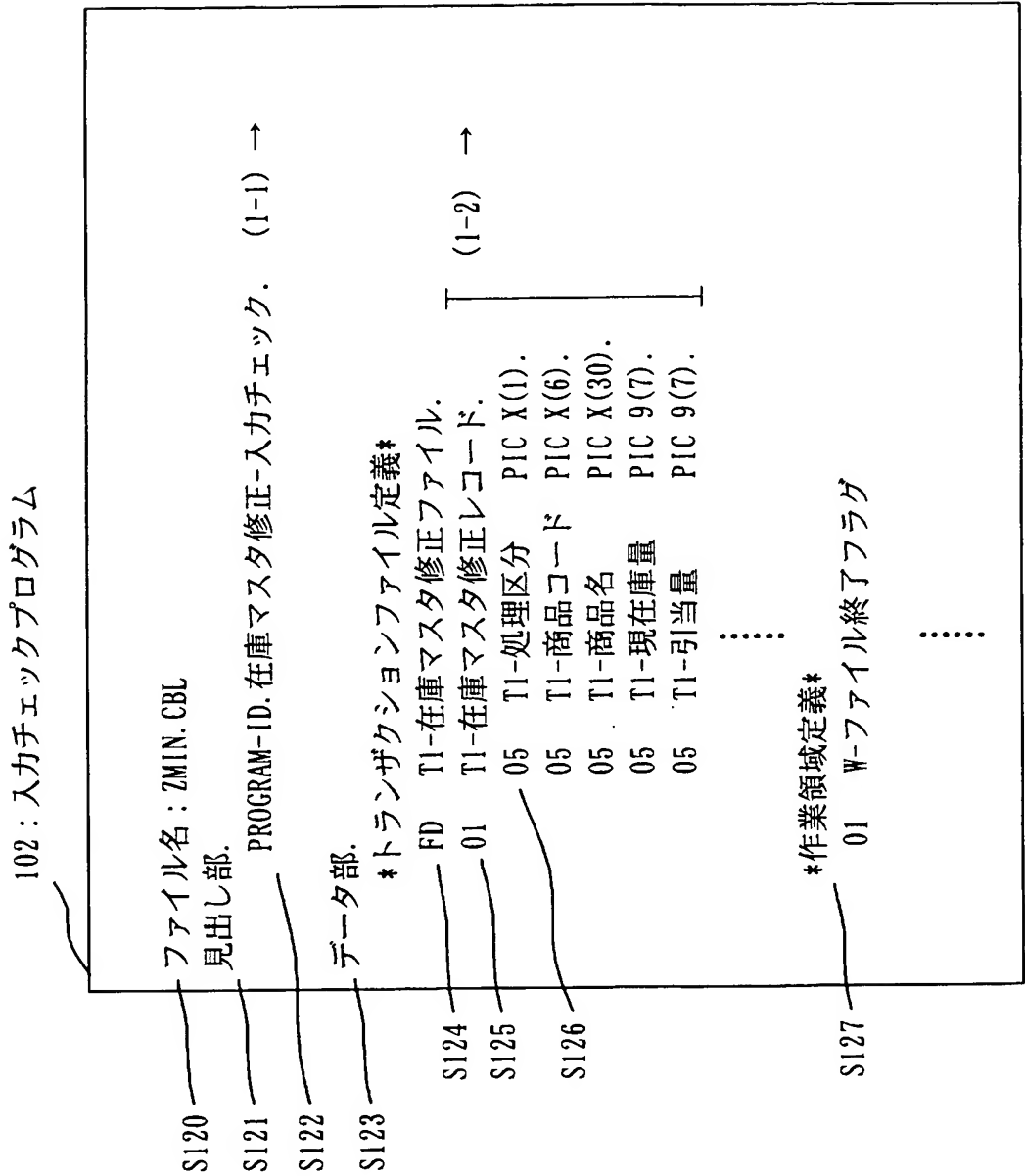
【図 15】

位置 番号	抽出する要素	変換先の位置	抽出時の特定のしかた	構造、構文上の変換	この抽出・変換の理由づけ
抽出元: 見出し部					
(1-1)	プログラム名	クラス名	PROGRAM-ID 段落から 抽出	・入力・チェックプログラムの 接辞をUICラスの接辞に置換 ・FILEクラスの接辞を付加 ・外部名は命名規則に従って FILEクラスのファイル名を挿 入する	
		環境部 FILEクラスに対するリ ボジトリ指定の内部名	"		
		クラス終わり見出し	"	・クラス名と同じ	
抽出元: データ部					
(1-2)	トランザクシヨ ン・レコードの定 義	クラス変数→WS節	トランザクシヨ・ファ イルの接辞が付いている		画面からの入力をレコードの形式 にしてFILEクラスに送る。
抽出元: 手続き部 → 変換先: UICラス、クラスメソッド、入力チェックメソッド					
(1-4)	主処理	入力チェックメソッドの 手続き部	前処理により既知	a. ファイルのOPEN、CLOSE 文を消す。 b. 変換処理コントロールへの PERFORM文から繰り返しの 指定 (UNTIL) を消す c. STOPRUNを EXIT METHODに置換する	a. UICラスでは入力は画面から 行い、チェック済みレコードは FILEクラスに出力する。 b. レコードを一件しか処理しな い。
(1-5)	変換処理コントロ ール	"	"	トランザクシヨン入力への PERFORM文を消す	レコード一件しか処理しないの で、次のトランザクシヨン・レコー ドを入力しない
(1-6)	トランザクシヨ ン入力	"	"	トランザクシヨン・ファイルの READ文をCONTINUE文に置換 する	画面からの入力データが直ちにト ランザクシヨン・レコードに入る ようになる
(1-7)	変換処理	"	"	トランザクシヨン・レコードから チェック済みトランザクシヨ ン・レコードへのMOVE文を消す	トランザクシヨン・レコードをチ ェックしてデータが正しければ、 そのままFILEクラスに送る
(1-8)	チェック済みトラ ンザクシヨン出力	"	"	チェック済みトランザクシヨ ン・ファイルへのWRITE文を消し、 FILEクラスの、マッピング更新 メソッドへのINVOKE文に置き 換える	チェックしたレコードはマッピン グ更新メソッドへの引数となる

【図 16】

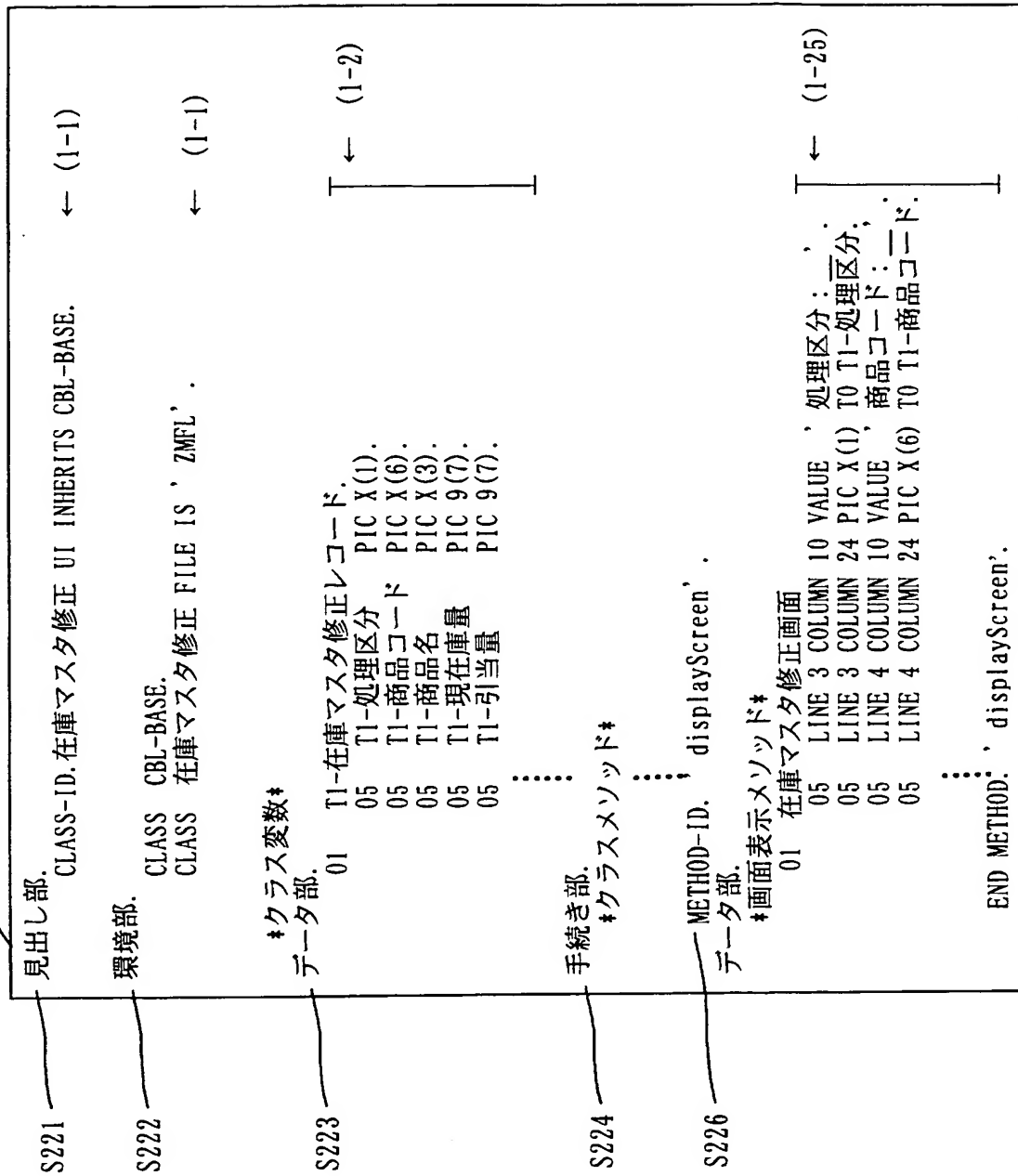
位置	抽出する要素	変換先の位置	抽出時の特定のしか た	構造、構文上の変換	この抽出・変換の理由づけ
(1-25)	伝票定義	画面表示メソッド 画面定義	レポート節でTYPE DETAIL (明細項目) の伝票項目を抽出	<p>a. 伝票の行位置、桁位置を 画面の行位置、桁位置に対 応付ける。</p> <p>b. 伝票項目のSOURCE指定 を画面項目のTO指定に置 き換える。</p> <p>c. トランザクションレコー ドに含まれるが、マスタレ コードに含まれないデー タ項目を元に画面項目を 生成する。</p>	c. 結果出力プログラムの、マ スタレコードの項目を出力 するが、画面からはトラン ザクションの項目を入力す る

【図 17】

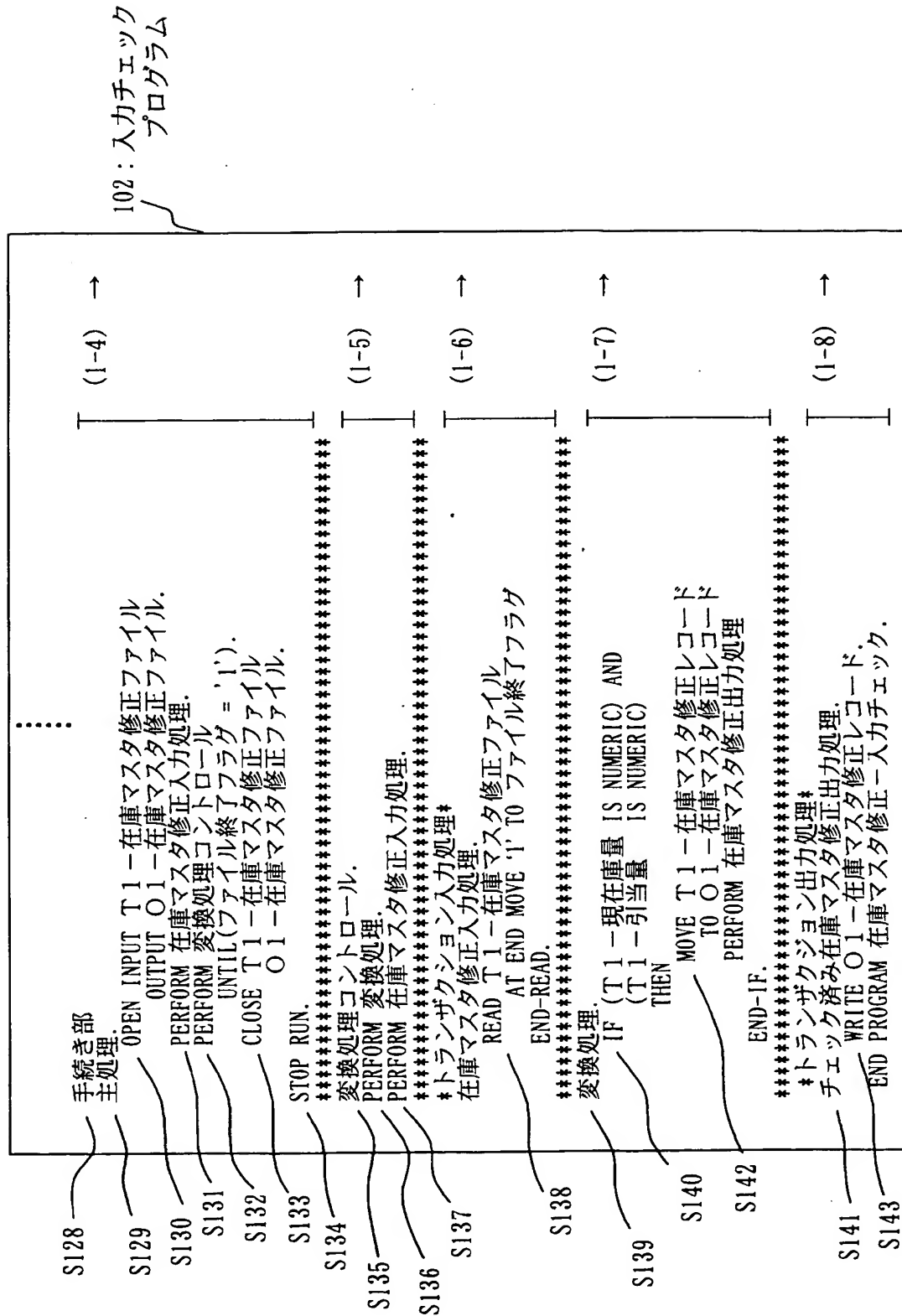


【図 18】

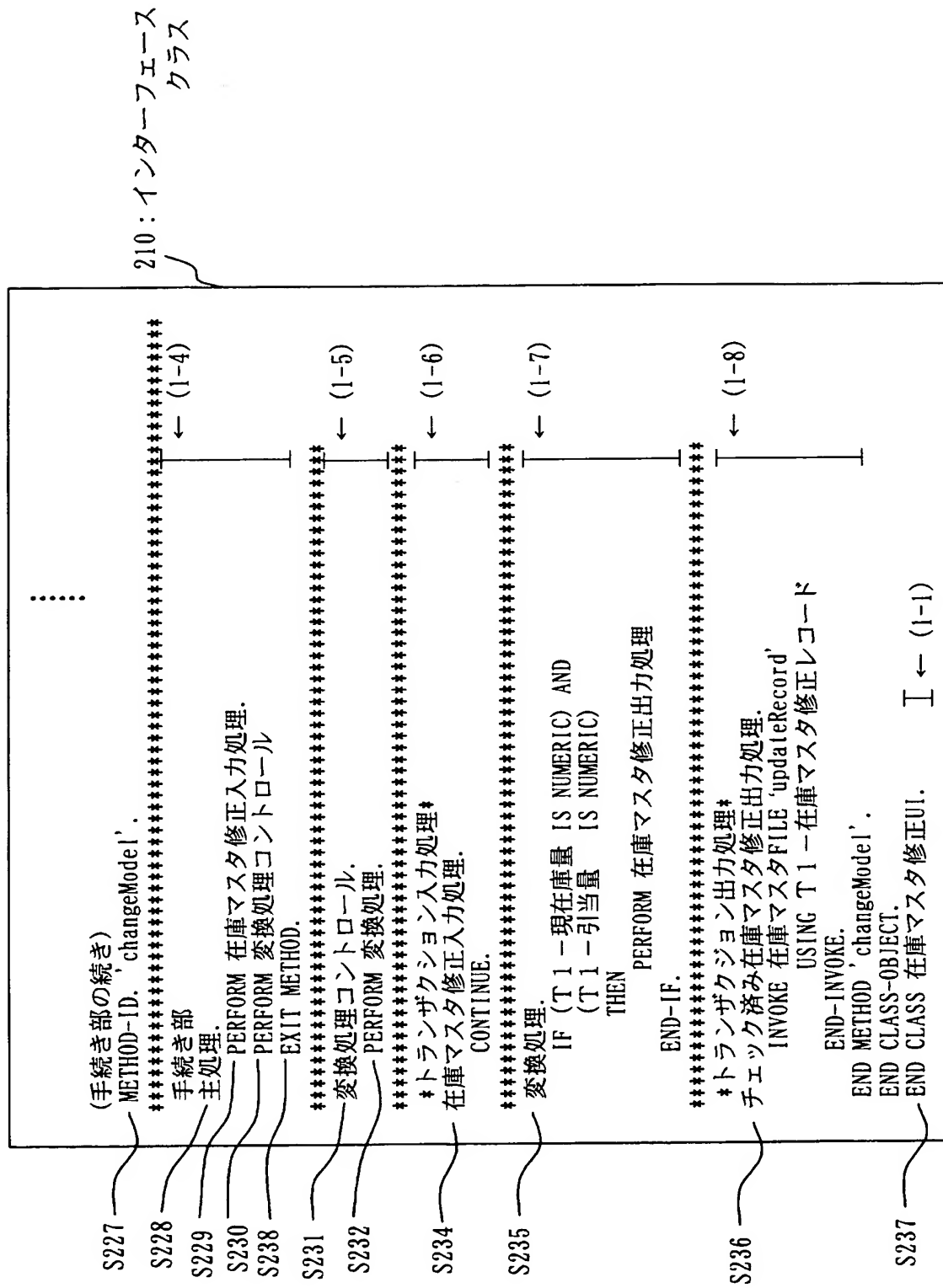
210: インターフェースクラス



【図 19】



【図 20】



【図 21】

1. 入力・チェックプログラム(変換前プログラム)

ファイル名: ZMIN.CBL

000001 IDENTIFICATION DIVISION.

000002 PROGRAM-ID. 在庫マスタ修正-入力チェック.

(1-1)→

000003 ENVIRONMENT DIVISION.

000004 INPUT-OUTPUT SECTION.

000005 FILE-CONTROL.

000006 SELECT T1-在庫マスタ修正ファイル ASSIGN TO SYS010-UT-MT-S.

000008 SELECT O1-在庫マスタ修正ファイル ASSIGN TO SYS020-DA-DK-S.

000010 DATA DIVISION.

000011 FILE SECTION.

000012 FD T1-在庫マスタ修正ファイル.

(1-2)→

000013 01 T1-在庫マスタ修正レコード.

000014 05 T1-処理区分 PIC X(1).

000015 05 T1-商品コード PIC X(6).

000016 05 T1-商品名 PIC X(30).

000017 05 T1-現在庫量 PIC 9(7).

000018 05 T1-引当量 PIC 9(7).

000019 05 T1-発注点 PIC 9(7).

000020 05 T1-基準在庫量 PIC 9(7).

000021 05 T1-仕入先コード PIC X(5).

000022 FD O1-在庫マスタ修正ファイル.

000023 01 O1-在庫マスタ修正レコード.

000024 05 O1-処理区分 PIC X(1).

000025 05 O1-商品コード PIC X(6).

000026 05 O1-商品名 PIC X(30).

000027 05 O1-現在庫量 PIC 9(7).

000028 05 O1-引当量 PIC 9(7).

000029 05 O1-発注点 PIC 9(7).

000030 05 O1-基準在庫量 PIC 9(7).

000031 05 O1-仕入先コード PIC X(5).

000036 PROCEDURE DIVISION.

000037 主処理.

(1-4)→

000038 OPEN INPUT T1-在庫マスタ修正ファイル

000039 OUTPUT O1-在庫マスタ修正ファイル.

000040 PERFORM 在庫マスタ修正入力処理.

000041 PERFORM 変換処理コントロール

000042 UNTIL(ファイル終了フラグ = '1').

000043 CLOSE T1-在庫マスタ修正ファイル

000044 O1-在庫マスタ修正ファイル.

000045 STOP RUN.

000046 *****

(1-5)→

000047 変換処理コントロール.

000048 PERFORM 変換処理.

000049 PERFORM 在庫マスタ修正入力処理.

000050 *****

000051 在庫マスタ修正入力処理.

(1-6)→

000052 READ T1-在庫マスタ修正ファイル

000053 AT END MOVE '1' TO ファイル終了フラグ

000054 END-READ.

000055 *****

(1-7)→

000056 変換処理.

000057 IF (T1-現在庫量 IS NUMERIC) AND

000058 (T1-引当量 IS NUMERIC)

000059 THEN

000060 MOVE T1-在庫マスタ修正レコード

000061 TO O1-在庫マスタ修正レコード

000062 PERFORM チェック済み在庫マスタ修正出力処理

000063 END-IF.

000064 *****

000065 チェック済み在庫マスタ修正出力処理.

(1-8)→

000066 WRITE O1-在庫マスタ修正レコード.

000067 END PROGRAM 在庫マスタ修正-入力チェック.

【図 22】

5. インターフェースクラス(中間プログラム)

ファイル名: ZMUI.CBL

000001 IDENTIFICATION DIVISION.

000002 CLASS-ID. 在庫マスタ修正 UI INHERITS CBL-BASE. [←(1-1)
(2-1)→

000003 ENVIRONMENT DIVISION.

000004 CONFIGURATION SECTION.

000005 REPOSITORY.

000006 CLASS CBL-BASE.

000007 CLASS 在庫マスタ修正 FILE IS 'ZMFL'. [←(1-1)

000008 IDENTIFICATION DIVISION.

000009 CLASS-OBJECT.

000010 DATA DIVISION.

000011*****

000012* クラス変数

000013*****

000014 WORKING-STORAGE SECTION.

000015 01 T1-在庫マスタ修正レコード.

000016 05 T1-処理区分 PIC X(1). [←(1-2)

000017 05 T1-商品コード PIC X(6). (2-2)→

000018 05 T1-商品名 PIC X(30).

000019 05 T1-現在庫量 PIC 9(7).

000020 05 T1-引当量 PIC 9(7).

000021 05 T1-発注点 PIC 9(7).

000022 05 T1-基準在庫量 PIC 9(7).

000023 05 T1-仕入先コード PIC X(5).

000024 01 終了フラグ PIC X(1).

000025 PROCEDURE DIVISION.

000026*****

000027* クラス・メソッド

000028*****

000029 IDENTIFICATION DIVISION.

000030 METHOD-ID. 'displayScreen'.

000031*****

000032 DATA DIVISION.

000033 SCREEN SECTION.

000034 01 在庫マスタ細目.

000035 05 LINE 3 COLUMN 10 VALUE '処理区分' : _____. [←(1-25)

000036 05 LINE 3 COLUMN 24 PIC X(1) TO T1-処理区分.

000037 05 LINE 4 COLUMN 10 VALUE '商品コード' : _____. (2-3)→

000038 05 LINE 4 COLUMN 24 PIC X(6) TO T1-商品コード.

000039 05 LINE 5 COLUMN 10 VALUE '商品名' : _____. (1-25)

000040 05 LINE 5 COLUMN 24 PIC X(30) TO T1-商品名.

000041 05 LINE 6 COLUMN 10 VALUE '現在庫量' : _____. (2-3)→

000042 05 LINE 6 COLUMN 24 PIC 9(7) TO T1-現在庫量.

000043 05 LINE 7 COLUMN 10 VALUE '引当量' : _____. (1-25)

000044 05 LINE 7 COLUMN 24 PIC 9(7) TO T1-引当量.

000045 05 LINE 8 COLUMN 10 VALUE '発注点' : _____. (2-3)→

000046 05 LINE 8 COLUMN 24 PIC 9(7) TO T1-発注点.

000047 05 LINE 9 COLUMN 10 VALUE '基準在庫量' : _____. (1-25)

000048 05 LINE 9 COLUMN 24 PIC 9(7) TO T1-基準在庫量.

000049 05 LINE 10 COLUMN 10 VALUE '仕入先コード' : _____. (2-3)→

000050 05 LINE 10 COLUMN 24 PIC X(5) TO T1-仕入先コード.

000051 05 LINE 11 COLUMN 40 VALUE '終了' : _____. (1-25)

000052 05 LINE 11 COLUMN 24 PIC X(1) TO 終了フラグ.

000053 PROCEDURE DIVISION.

000054 DISPLAY 在庫マスタ細目.

000055 ACCEPT 在庫マスタ細目.

000056 EXIT METHOD.

000057 END METHOD. 'displayScreen'.

000058

:

【図 23】

(5. インターフェースクラスの続き)

```

000059*****
000060 IDENTIFICATION DIVISION.
000061 METHOD-ID. 'uiMain'.
000062*****
000063 PROCEDURE DIVISION.
000064     PERFORM UNTIL (終了フラグ = '1')
000065         INVOKE SELF 'displayScreen'
000066         INVOKE SELF 'changeModel'
000067     END-PERFORM.
000068     EXIT METHOD.
000069 END METHOD 'uiMain'.
000070
000071*****
000072 IDENTIFICATION DIVISION.
000073 METHOD-ID. 'changeModel'.
000074*****
000075 PROCEDURE DIVISION.
000076 主処理.
000077     PERFORM 在庫マスタ修正入力処理.
000078     PERFORM 変換処理コントロール
000079     EXIT METHOD.
000080*****
000081 変換処理コントロール.
000082     PERFORM 変換処理.
000083*****
000084 在庫マスタ修正入力処理.
000085     CONTINUE.
000086*****
000087 変換処理.
000088     IF (T1-現在庫量 IS NUMERIC) AND
000089         (T1-引当量 IS NUMERIC)
000090         THEN
000091             PERFORM チェック済み在庫マスタ修正出力処理
000092         END-IF.
000093*****
000094 チェック済み在庫マスタ修正出力処理.
000095     INVOKE 在庫マスタ FILE 'updateRecord'
000096         USING T1-在庫マスタ修正レコード
000097     END-INVOKE.
000098 END METHOD 'changeModel'.
000099 END CLASS-OBJECT.
000100 END CLASS 在庫マスタ修正 UI.

```

(2-4) →
 ← (1-4)
 (2-5) →
 ← (1-5)
 (2-6) →
 ← (1-6)
 (2-7) →
 ← (1-7)
 (2-8) →
 ← (1-8)
 (2-9) →
 ← (1-1)

【図 2 4】

4. 結果出力プログラム

```
ファイル名: ZMOT.CBL
000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. 在庫マスタ修正 - 結果出力.
000003 ENVIRONMENT DIVISION.
000004 INPUT-OUTPUT SECTION.
000005 FILE-CONTROL.
000006     SELECT T1 - 在庫マスタファイル
000007         ASSIGN TO SYS010-DA-DK-S.
000008     SELECT O1 - 出力ファイル
000009         ASSIGN TO SYS020-UR-LP-S.
000010 DATA DIVISION.
000011 FILE SECTION.
000012 FD  T1 - 在庫マスタファイル BLOCK CONTAINS 35 RECORDS.
000013 01  T1 - 在庫マスタレコード.
000014     05  T1 - 商品コード PIC X(6).
000015     05  T1 - 商品名 PIC X(30).
000016     05  T1 - 現在在庫量 PIC 9(7).
000017     05  T1 - 引当量 PIC 9(7).
000018     05  T1 - 発注点 PIC 9(7).
000019     05  T1 - 基準在庫量 PIC 9(7).
000020     05  T1 - 仕入先コード PIC X(5).
000021 FD  O1 - 出力ファイル LABEL RECORD OMITTED
000022         REPORT IS 在庫マスタリスト.
000023 WORKING-STORAGE SECTION.
000024 01  ファイル終了フラグ PIC X VALUE '0'.
000025 01  W - 日付 PIC X(8).
000026 REPORT SECTION.
000027 RD  在庫マスタリスト CONTROLS ARE T1 - 商品コード
000028         PAGE LIMITS 66 LINES
000029         HEADING 1
000030         FIRST DETAIL 7
000031         LAST DETAIL 56
000032         FOOTING 66.
000033 01  在庫マスタ細目 TYPE IS DETAIL.
000034     05  LINE NUMBER IS 4.
000035         10  COLUMN 10 PIC X(12) VALUE '商品コード: '.
000036         10  COLUMN 22 PIC X(6) SOURCE T1 - 商品コード.
000037     05  LINE NUMBER IS 5.
000038         10  COLUMN 10 PIC X(8) VALUE '商品名: '.
000039         10  COLUMN 18 PIC X(30) SOURCE T1 - 商品名.
000040     05  LINE NUMBER IS 6.
000041         10  COLUMN 10 PIC X(10) VALUE '現在在庫量: '.
000042         10  COLUMN 20 PIC 9(7) SOURCE T1 - 現在在庫量.
000043     05  LINE NUMBER IS 7.
000044         10  COLUMN 10 PIC X(8) VALUE '引当量: '.
000045         10  COLUMN 18 PIC 9(7) SOURCE T1 - 引当量.
000046     05  LINE NUMBER IS 8.
000047         10  COLUMN 10 PIC X(8) VALUE '発注点: '.
000048         10  COLUMN 18 PIC 9(7) SOURCE T1 - 発注点.
000049     05  LINE NUMBER IS 9.
000050         10  COLUMN 10 PIC X(12) VALUE '基準在庫量: '.
000051         10  COLUMN 22 PIC 9(7) SOURCE T1 - 基準在庫量.
000052     05  LINE NUMBER IS 10.
000053         10  COLUMN 10 PIC X(14) VALUE '仕入先コード: '.
000054         10  COLUMN 24 PIC X(5) SOURCE T1 - 仕入先コード.
000055 01  TYPE IS PAGE HEADING.
000056     05  LINE NUMBER IS 2.
000057         10  COLUMN 41 PIC X(18) VALUE '*** 在庫マスタ'.
000058         10  COLUMN 60 PIC X(16) VALUE '伝票 ***'.
000059         10  COLUMN 95 PIC X(8) SOURCE W - 日付.
000060 01  TYPE IS CONTROL FOOTING T1 - 商品コード
000061     NEXT GROUP NEXT PAGE.
```

(1-25)→

⋮

【図 25】

(4. 結果出力プログラムの続き)

```
000062 PROCEDURE DIVISION.  
000063 主処理.  
000064     OPEN INPUT T1-在庫マスタファイル  
000065         OUTPUT O1-出力ファイル.  
000066     MOVE CURRENT-DATE TO W-日付.  
000067     INITIATE 在庫マスタリスト.  
000068     PERFORM 在庫マスタファイル入力処理.  
000069     PERFORM 報告書作成コントロール  
000070         UNTIL (ファイル終了フラグ = '1').  
000071     TERMINATE 在庫マスタリスト.  
000072     CLOSE T1-在庫マスタファイル  
000073         O1-出力ファイル.  
000074     STOP RUN.  
000075*****  
000076 報告書作成コントロール.  
000077     GENERATE 在庫マスタ細目.  
000078     PERFORM 在庫マスタファイル入力処理.  
000079*****  
000080 在庫マスタファイル入力処理.  
000081     READ T1-在庫マスタファイル  
000082         AT END MOVE '1' TO ファイル終了フラグ  
000083     END-READ.  
000084 END PROGRAM 在庫マスタ修正-結果出力.
```

【図 26】

位置 番号	抽出する要素	変換先の位置	抽出時の特定のしかた	構造、構文上の変換	この抽出・変換の理由づけ
抽出元:見出し部					
(1-10)	プログラム名	クラス名	PROGRAM-ID段落から 抽出	マッチングプログラムの接辞を ファイルクラスの接辞に置換	
		クラス終わり見出し	"	"	
抽出元:環境部					
(1-11)	旧マスタファイル のSELECT文	環境部	ファイル命名規則から 旧マスタ・ファイルの名 前を判断, SELECT文を 抽出	a. ORGANIZATION指定を INDEXED (索引編成) にする. b. ACCESS RANDOM指定を付 加する. c. RECORD KEY指定を付加す る.	
抽出元:データ部					
(1-13)	旧マスタ・レコー ド定義	クラス変数→ファイル節	命名規則から旧マスタ ファイルの名前を判断		
(1-12)	トランザクション ・レコード定義	マッチング更新メソッド →データ部LINKAGE節	命名規則からトランザ クション・ファイルの名 前を判断		ファイルからREADだったのが、 メソッドへの引数になる
		マッチング更新メソッド →手続き部見出しの USING指定	"	レコード名だけを挿入	トランザクション・レコードを引 数として使う
(1-14)	フラグ項目	マッチング更新メソッド →データ部WS節	WS節のデータをそのま ま抽出		マッチング更新プログラムが持つ ていたロジック (トランザクショ ン・レコードとマスタ・レコードを 入力、キーの比較) をそのまま活 用するので、フラグ項目も必要に なる

【図 27】

抽出元: データ部 → 変換先: マッチング更新メソッド, 手続き部			
主処理	マッチング更新メソッド の手続き部	前処理により既知	a. OPEN文とCLOSE文から、トランザクション、新 マスタ・ファイルの指定を消す、旧マスタファ イルのOPENモードをI-Oに変える。 b. 更新コントリール処理へのPERFORM文から、繰 り返しの指定 (UNTIL) を消す。 c. STOP RUN文を EXIT METHOD文に変える。
(1-15)			a. トランザクションは引数として入力する。 マスタ・ファイルはREADもWRITEも同一のファ イルに対して行う。 b. レコード一件しか処理しない
(1-16)	更新コントリール	"	マスタ・レコードを一件しか処理しないので、マ スタ処理 (次のマスタ・レコードを準備する) は 要らなくなる
(1-17)	合致処理	"	トランザクション・レコードを一件しか処理し ないので、次のトランザクションを準備する処 理を呼び出さない
(1-18)	トランザクション処 理	"	バッチ処理 (順ファイル) では 更新も追加もWRITEだったが、 一件別処理 (索引ファイル) では追加: WRITE, 更新: REWRITEで分ける必要がある
(1-19, 更新処理, (1-20, 追加処理) は抽出してそのまま挿入			
(1-21)	新マスタ出力	"	WRITE文をREWRITEに変える。
(1-22)	削除処理	"	節の名前を追加新マスタ出力処理に変える。 旧マスタ入力へのPERFORM文を消して、マスタ・レ コードのDELETE文に置き換える。
(1-23)	トランザクション入 力	"	a. READ文を消す。 b. トランザクション・レコードのキーからMOVE しているMOVE文だけ抽出して変換先に挿入する
(1-24, 旧マスタ入力) は抽出してそのまま挿入			

【図 28】

位置 番号	抽出する要素	変換先の位置	抽出時の特定の しかた	構造、構文上の変換	この抽出・変換の 理由づけ
(1-9)	手続き部 SORT文の ASCENDING 指定にあるデ ータ項目名	環境部 旧マスタファイルの SELECT文、レコード キー指定	単純検索	トランザクション・レコード の接辞を旧マスタ・レコード の接辞に変える	

【図29】

3. マッチング更新プログラム(変換前プログラム)

プログラム名: ZMUP.CBL

```

000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. 在庫マスタ修正 - マッチング更新.
000003 ENVIRONMENT DIVISION.
000004 INPUT-OUTPUT SECTION.
000005 FILE-CONTROL.
000006     SELECT T1 - 在庫マスタ修正ファイル
000007             ASSIGN TO SYS010-DA-DK-S
000008             ORGANIZATION LINE SEQUENTIAL.
000009     SELECT M1 - 在庫マスタファイル
000010             ASSIGN TO SYS030-DA-DK-S
000011             ORGANIZATION LINE SEQUENTIAL.
000012     SELECT M2 - 在庫マスタファイル
000013             ASSIGN TO SYS031-DA-DK-S
000014             ORGANIZATION LINE SEQUENTIAL.
000015 DATA DIVISION.
000016 FILE SECTION.
000017 FD      T1 - 在庫マスタ修正ファイル.
000018 01      T1 - 在庫マスタ修正レコード.
000019 05      T1 - 処理区分          PIC X(1).
000020 88      T1 - 追加 VALUE '1'.
000021 88      T1 - 更新 VALUE '2'.
000022 88      T1 - 削除 VALUE '9'.
000023 05      T1 - 商品コード      PIC X(6).
000024 05      T1 - 商品名          PIC X(30).
000025 05      T1 - 現在在庫量      PIC 9(7).
000026 05      T1 - 引当量          PIC 9(7).
000027 05      T1 - 発注点          PIC 9(7).
000028 05      T1 - 基準在庫量      PIC 9(7).
000029 05      T1 - 仕入先コード  PIC X(5).
000030 FD      M1 - 在庫マスタファイル.
000031 01      M1 - 在庫マスタレコード.
000032 05      M1 - 商品コード      PIC X(6).
000033 05      M1 - 商品名          PIC X(30).
000034 05      M1 - 現在在庫量      PIC 9(7).
000035 05      M1 - 引当量          PIC 9(7).
000036 05      M1 - 発注点          PIC 9(7).
000037 05      M1 - 基準在庫量      PIC 9(7).
000038 05      M1 - 仕入先コード  PIC X(5).
000039 FD      M2 - 在庫マスタファイル.
000040 01      M2 - 在庫マスタレコード.
000041 05      M2 - 商品コード      PIC X(6).
000042 05      M2 - 商品名          PIC X(30).
000043 05      M2 - 現在在庫量      PIC 9(7).
000044 05      M2 - 引当量          PIC 9(7).
000045 05      M2 - 発注点          PIC 9(7).
000046 05      M2 - 基準在庫量      PIC 9(7).
000047 05      M2 - 仕入先コード  PIC X(5).

```

:

【図 30】

(3. マッチング更新プログラムの続き(1))

```

000048
000049 WORKING-STORAGE SECTION.
000050 01 作業領域.
000051     05 W-トランザクションキー PIC X(6) VALUE LOW-VALUE.
000052     05 W-マスタキー PIC X(6).
000053 01 フラグ.
000054     05 ファイル終了フラグ1 PIC X VALUE '0'.
000055     05 ファイル終了フラグ2 PIC X VALUE '0'.
000056     05 処理終了フラグ PIC X VALUE '0'.
000057
000058 PROCEDURE DIVISION.
000059 主処理.
000060     OPEN INPUT T1-在庫マスタ修正ファイル
000061             M1-在庫マスタファイル
000062     OUTPUT M2-在庫マスタファイル.
000063     PERFORM 在庫マスタ修正入力処理.
000064     PERFORM 旧在庫マスタ入力処理.
000065     PERFORM 更新コントロール処理
000066             UNTIL (処理終了フラグ = '1').
000067     CLOSE T1-在庫マスタ修正ファイル
000068             M1-在庫マスタファイル
000069             M2-在庫マスタファイル.
000070     STOP RUN.
000071 *****
000072 更新コントロール処理.
000073     IF (W-トランザクションキー = W-マスタキー)
000074     THEN
000075         PERFORM 合致処理.
000076     ELSE
000077         IF (W-トランザクションキー > W-マスタキー)
000078         THEN
000079             PERFORM マスタ処理
000080         ELSE
000081             PERFORM トランザクション処理
000082         END-IF
000083     END-IF.
000084 *****
000085 合致処理.
000086     IF (W-トランザクションキー = HIGH-VALUE)
000087     THEN
000088         MOVE '1' TO 処理終了フラグ
000089     ELSE
000090         EVALUATE T1-処理区分
000091             WHEN T1-更新
000092                 PERFORM 更新処理
000093                 PERFORM 新在庫マスタ出力処理
000094                 PERFORM 旧在庫マスタ入力処理
000095             WHEN T1-削除
000096                 PERFORM 削除処理
000097         END-EVALUATE
000098     PERFORM 在庫マスタ修正入力処理
000099     END-IF.
000100 *****

```

⋮

【図 3 1】

(3. マッチング更新プログラムの続き(2))

```
000101 マスタ処理.
000102     MOVE M1-在庫マスタレコード
000103         TO M2-在庫マスタレコード.
000104     PERFORM 新在庫マスタ出力処理.
000105     PERFORM 旧在庫マスタ入力処理.
000106*****
000107 トランザクション処理. (1-18) →
000108     IF T1-追加
000109         THEN
000110             PERFORM 追加処理
000111             PERFORM 新在庫マスタ出力処理.
000112     END-IF.
000113     PERFORM 在庫マスタ修正入力処理.
000114*****
000115 更新処理. (1-19) →
000116     MOVE T1-商品名      TO M2-商品名.
000117     MOVE T1-現在庫量    TO M2-現在庫量.
000118     MOVE T1-引当量      TO M2-引当量.
000119     MOVE T1-発注点      TO M2-発注点.
000120     MOVE T1-基準在庫量  TO M2-基準在庫量.
000121     MOVE T1-仕入先コード TO M2-仕入先コード.
000122*****
000123 追加処理. (1-20) →
000124     MOVE T1-商品コード  TO M2-商品コード.
000125     MOVE T1-商品名      TO M2-商品名.
000126     MOVE T1-現在庫量    TO M2-現在庫量.
000127     MOVE T1-引当量      TO M2-引当量.
000128     MOVE T1-発注点      TO M2-発注点.
000129     MOVE T1-基準在庫量  TO M2-基準在庫量.
000130     MOVE T1-仕入先コード TO M2-仕入先コード.
000131*****
000132 新在庫マスタ出力処理. (1-21) →
000133     WRITE M2-在庫マスタレコード
000134     INVALID MOVE '1' TO 処理終了フラグ.
000135*****
000136 削除処理. (1-22) →
000137     PERFORM 旧在庫マスタ入力処理.
000138*****
000139 在庫マスタ修正入力処理. (1-23) →
000140     READ T1-在庫マスタ修正ファイル
000141         AT END MOVE '1' TO ファイル終了フラグ1
000142     END-READ.
000143     IF (ファイル終了フラグ1 = '1')
000144         THEN
000145             MOVE HIGH-VALUE TO W-トランザクションキー
000146         ELSE
000147             MOVE T1-商品コード
000148                 TO W-トランザクションキー
000149         END-IF.
000150*****
000151 旧在庫マスタ入力処理. (1-24) →
000152     READ M1-在庫マスタファイル
000153         AT END MOVE '1' TO ファイル終了フラグ2
000154     END-READ.
000155     IF (ファイル終了フラグ2 = '1')
000156         THEN
000157             MOVE HIGH-VALUE TO W-マスタキー
000158         ELSE
000159             MOVE M1-商品コード TO W-マスタキー
000160         END-IF.
000161 END PROGRAM 在庫マスタ修正-マッチング更新.
```

【図 3 2】

2. ソートプログラム(変換前プログラム)

ファイル名: ZMSR.CBL

000001 IDENTIFICATION DIVISION.

000002 PROGRAM-ID. 在庫マスタ修正 - ソート.

000003 ENVIRONMENT DIVISION.

000004 INPUT-OUTPUT SECTION.

000005 FILE-CONTROL.

000006 SELECT T1 - 在庫マスタ修正ファイル

000007 ASSIGN TO SYS010-UT-MT-S.

000008 SELECT O1 - 在庫マスタ修正ファイル

000009 ASSIGN TO SYS020-DA-DK-S.

000010 SELECT S1 - 中間ファイル

000011 ASSIGN TO WORK001.

000012 DATA DIVISION.

000013 FILE SECTION.

000014 FD T1 - 在庫マスタ修正ファイル.

000015 01 PIC X(70)

000016 FD O1 - 在庫マスタ修正ファイル.

000017 01 PIC X(70).

000018 SD S1 - 中間ファイル.

000019 01 S1 - 中間ファイルレコード.

000020 05 S1 - 処理区分 PIC X(1).

000021 05 S1 - 商品コード PIC X(6).

000022 05 S1 - 商品名 PIC X(30).

000023 05 S1 - 現在庫量 PIC 9(7).

000024 05 S1 - 引当量 PIC 9(7).

000025 05 S1 - 発注点 PIC 9(7).

000026 05 S1 - 基準在庫量 PIC 9(7).

000027 05 S1 - 仕入先コード PIC X(5).

000028 PROCEDURE DIVISION.

000029 SORT 中間ファイル

000030 ON ASCENDING KEY S1 - 商品コード

000031 USING T1 - 在庫マスタ修正ファイル

000032 GIVING O1 - 在庫マスタ修正ファイル.

000033 STOP RUN.

000034 END PROGRAM 在庫マスタ修正 - ソート.

I (1-9) →

【図 3 3】

6. ファイルクラス(中間プログラム)

ファイル名: ZMFL.CBL

000001 IDENTIFICATION DIVISION.

000002 CLASS-ID. 在庫マスタ修正 FILE INHERITS CBL-BASE.

← (1-10)
(2-10)→

000003 ENVIRONMENT DIVISION.

000004 CONFIGURATION SECTION.

000005 REPOSITORY.

000006 CLASS CBL-BASE.

000007 INPUT-OUTPUT SECTION.

000008 FILE-CONTROL.

000009 SELECT M1-在庫マスタファイル

000010 ASSIGN TO SYS030-DA-DK-I

000011 ORGANIZATION INDEXED

000012 ACCESS RANDOM

000013 RECORD KEY M1-商品コード.

← (1-11)
← (1-9)
(2-11)→

000014

000015 IDENTIFICATION DIVISION.

000016 CLASS-OBJECT.

000017 DATA DIVISION.

000018*****

000019* クラス変数

000020*****

000021 FILE SECTION.

000022 FD M1-在庫マスタファイル.

000023 01 M1-在庫マスタレコード.

000024 05 M1-商品コード PIC X(6).

000025 05 M1-商品名 PIC X(30).

000026 05 M1-現在在庫量 PIC S9(7).

000027 05 M1-引当量 PIC S9(7).

000028 05 M1-発注点 PIC S9(7).

000029 05 M1-基準在庫量 PIC S9(7).

000030 05 M1-仕入先コード PIC X(5).

000031

000032 PROCEDURE DIVISION.

000033*****

000034* クラス・メソッド

000035*****

000036 IDENTIFICATION DIVISION.

000037 METHOD-ID. 'updateRecord'.

000038*****

000039 DATA DIVISION.

000040 LINKAGE SECTION.

000041 01 T1-在庫マスタ修正レコード.

000042 05 T1-処理区分 PIC X(1).

000043 88 T1-追加 VALUE '1'.

000044 88 T1-更新 VALUE '2'.

000045 88 T1-削除 VALUE '9'.

000046 05 T1-商品コード PIC X(6).

000047 05 T1-商品名 PIC X(30).

000048 05 T1-現在在庫量 PIC 9(7).

000049 05 T1-引当量 PIC 9(7).

000050 05 T1-発注点 PIC 9(7).

000051 05 T1-基準在庫量 PIC 9(7).

000052 05 T1-仕入先コード PIC X(5).

000053

000054 WORKING-STORAGE SECTION.

000055 01 作業領域.

000056 05 W-トランザクションキー PIC X(6) VALUE LOW-VALUE.

000057 05 W-マスタキー PIC X(6).

000058 01 フラグ.

000059 05 ファイル終了フラグ1 PIC X VALUE '0'.

000060 05 ファイル終了フラグ2 PIC X VALUE '0'.

000061 05 処理終了フラグ PIC X VALUE '0'.

000062

000063 PROCEDURE DIVISION USING T1-在庫マスタ修正レコード.

000064 主処理.

000065 OPEN I-O M1-在庫マスタファイル.

000066 PERFORM 在庫マスタ修正入力処理.

000067 PERFORM 旧在庫マスタ入力処理.

000068 PERFORM 更新コントロール処理.

000069 CLOSE M1-在庫マスタファイル.

000070 EXIT METHOD.

:

← (1-14)

← (1-12)

← (1-15)

(2-14)→

← (1-13)
(2-12)→← (1-12)
(2-13)→

:

【図 3 4】

(6. ファイルクラスの続き)

```

000071*****
000072 更新コントロール処理.
000073 IF (W-トランザクションキー = W-マスタキー)
000074     THEN
000075         PERFORM 合致処理
000076     ELSE
000077         PERFORM トランザクション処理
000078     END-IF.
000079*****
000080 合致処理.
000081 IF (W-トランザクションキー = HIGH-VALUE)
000082     THEN
000083         MOVE '1' TO 処理終了フラグ
000084     ELSE
000085         EVALUATE T1-処理区分
000086             WHEN T1-更新
000087                 PERFORM 更新処理
000088                 PERFORM 新在庫マスタ出力処理
000089             WHEN T1-削除
000090                 PERFORM 削除処理
000091         END-EVALUATE
000092     END-IF.
000093*****
000094 トランザクション処理.
000095 IF T1-追加
000096     THEN
000097         PERFORM 追加処理
000098         PERFORM 追加新在庫マスタ出力処理.
000099     END-IF.
000100*****
000101 更新処理.
000102 MOVE T1-商品名      TO M1-商品名.
000103 MOVE T1-現在在庫量 TO M1-現在在庫量.
000104 MOVE T1-引当量     TO M1-引当量.
000105 MOVE T1-発注点     TO M1-発注点.
000106 MOVE T1-基準在庫量 TO M1-基準在庫量.
000107 MOVE T1-仕入先コード TO M1-仕入先コード.
000108*****
000109 追加処理.
000110 MOVE T1-商品コード TO M1-商品コード.
000111 MOVE T1-商品名     TO M1-商品名.
000112 MOVE T1-現在在庫量 TO M1-現在在庫量.
000113 MOVE T1-引当量     TO M1-引当量.
000114 MOVE T1-発注点     TO M1-発注点.
000115 MOVE T1-基準在庫量 TO M1-基準在庫量.
000116 MOVE T1-仕入先コード TO M1-仕入先コード.
000117*****
000118 追加新在庫マスタ出力処理.
000119 WRITE M1-在庫マスタレコード
000120     INVALID MOVE '1' TO 処理終了フラグ.
000121*****
000122 新在庫マスタ出力処理.
000123 REWRITE M1-在庫マスタレコード
000124     INVALID MOVE '1' TO 処理終了フラグ.
000125*****
000126 削除処理.
000127 DELETE M1-在庫マスタレコード
000128     INVALID MOVE '1' TO 処理終了フラグ.
000129*****
000130 在庫マスタ修正入力処理.
000131 MOVE T1-商品コード
000132     TO W-トランザクションキー M1-商品コード
000133*****
000134 旧在庫マスタ入力処理.
000135 READ M1-在庫マスタファイル
000136     INVALID MOVE '1' TO ファイル終了フラグ2.
000137 IF (ファイル終了フラグ2 = '1')
000138     THEN
000139         MOVE HIGH-VALUE TO W-マスタキー
000140     ELSE
000141         MOVE M1-商品コード TO W-マスタキー
000142     END-IF.
000143 END METHOD 'updateRecord'.
000144 END CLASS-OBJECT.
000145 END CLASS 在庫マスタ修正 FILE.

```

(2-14 つづき) →

← (1-16)

← (1-17)

← (1-18)

← (1-19)

← (1-20)

← (1-21)

← (1-21)

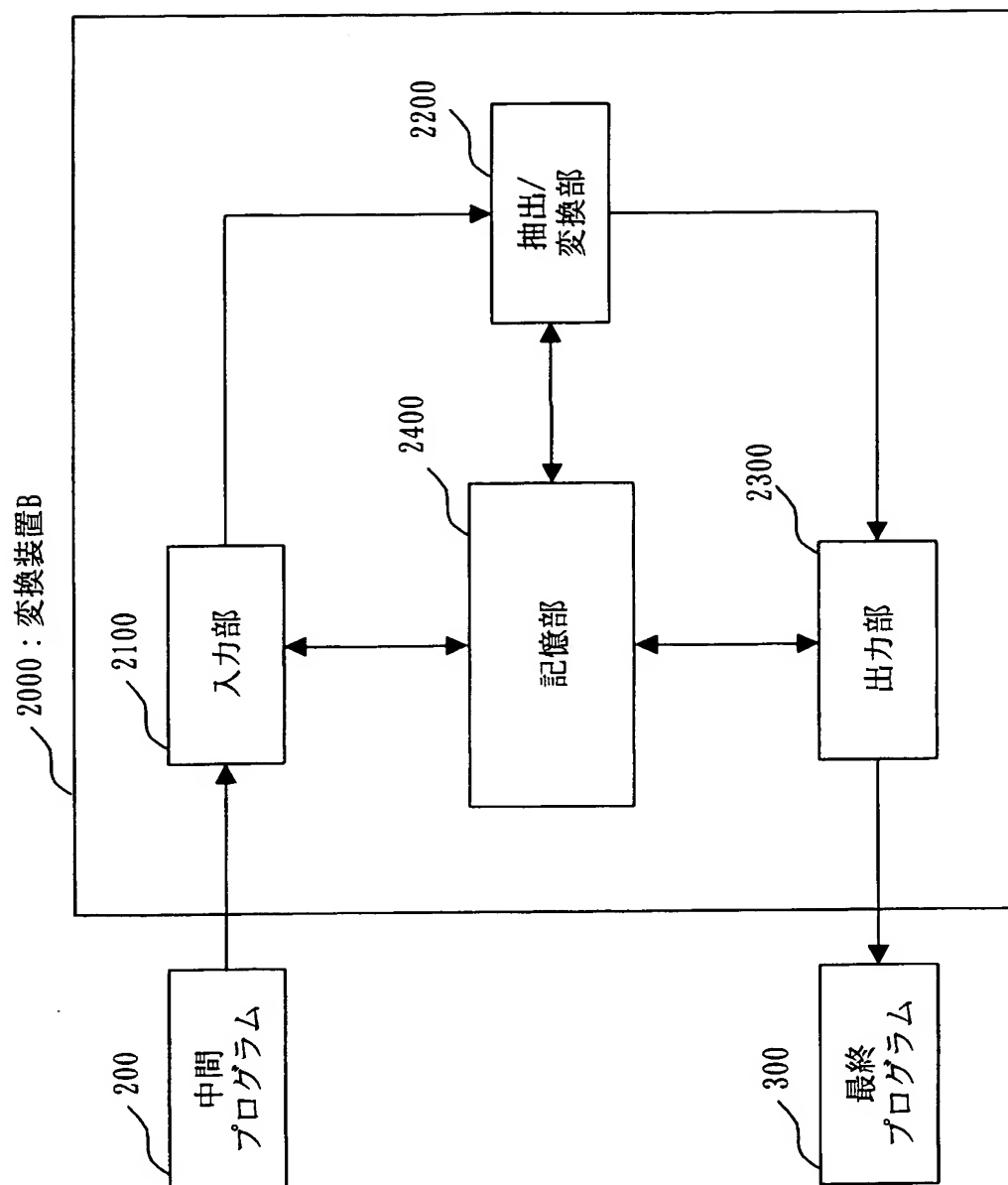
← (1-22)

← (1-23)

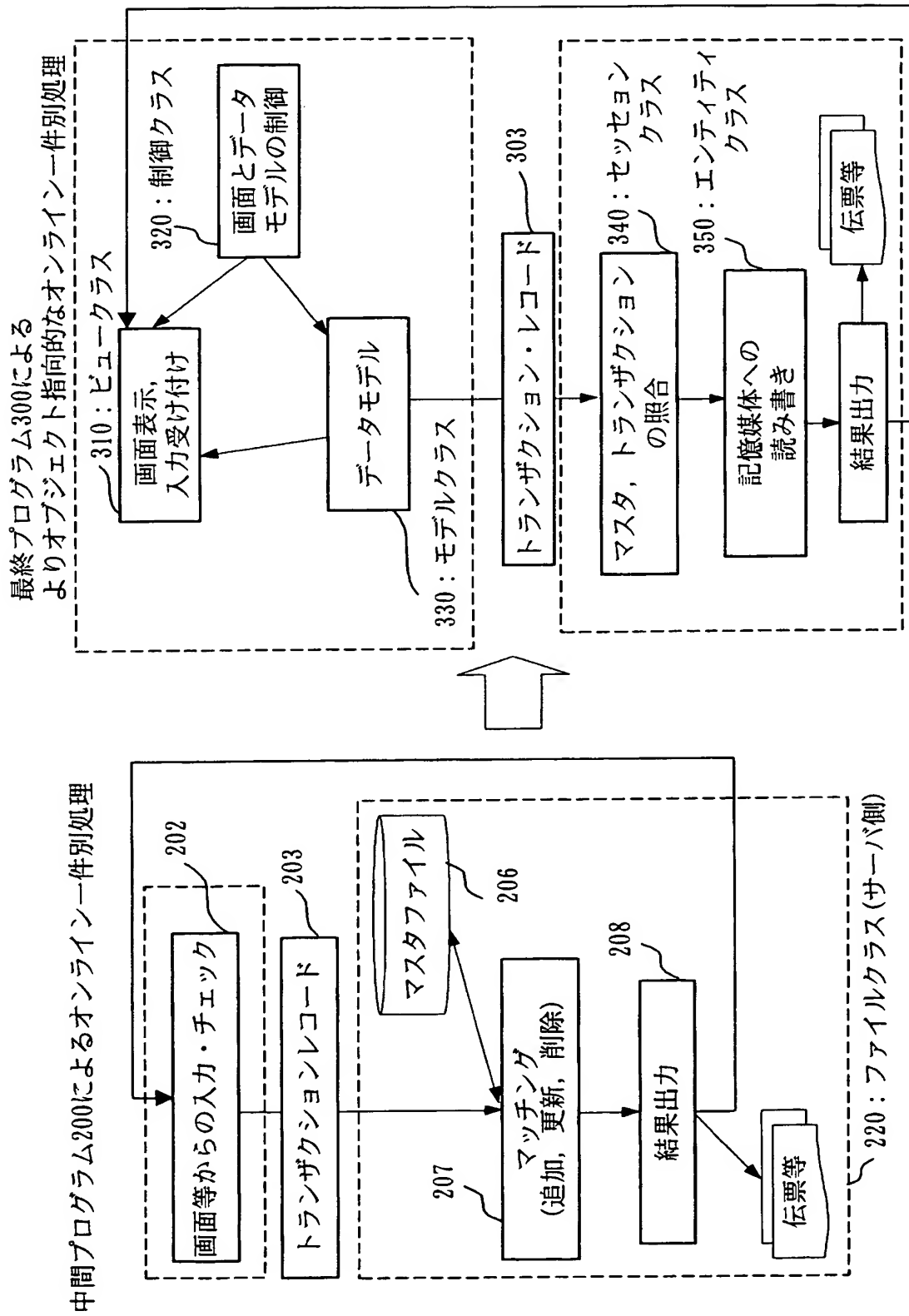
← (1-24)

← (1-10)

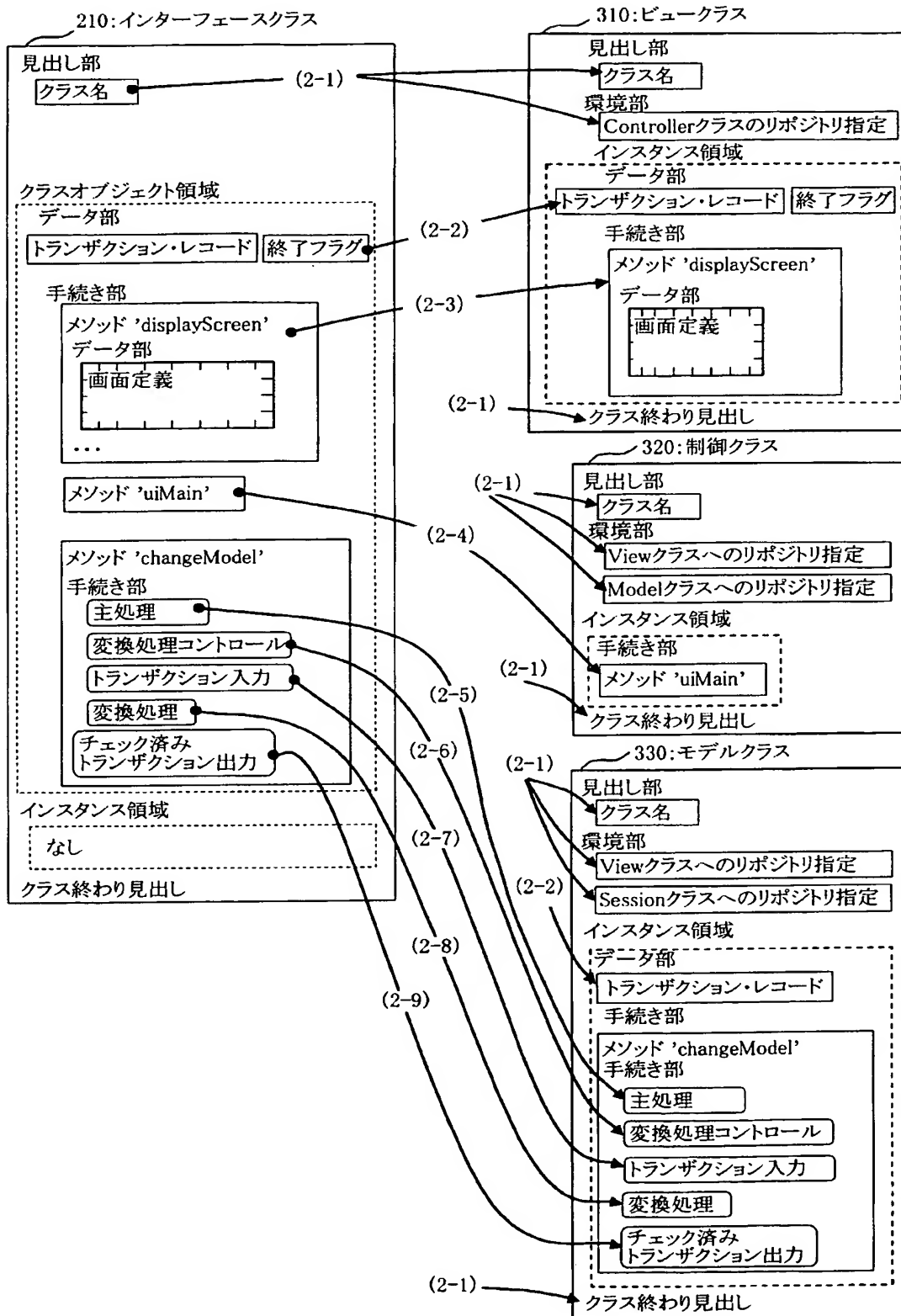
【図 35】



【図 36】



【図 37】



【図 38】

番号	抽出する要素	変換先の位置	構造、構文上の変換	この抽出・変換の理由づけ
抽出元: UI, クラス見出し部				
(2-1)	クラス名	クラス名 環境部 Controller クラスに対する リポジトリ指定の内部名	View クラスの接辞を付加 ・ Controller クラス名の接辞を付加 ・ 外部名は命名規則に従い Controller クラスのファイル名を 挿入する.	
		クラス終わり見出し	View クラスの接辞を付加	
抽出元: UI, クラス変数				
(2-2)	トランザクシ ョン・ レコードと, 終了フラグ	インスタンス変数→WS 節	データ項目に PROPERTY 指定を 付加	画面からの入力をレコードの形式に格納 する. あとで Model クラスからこのレコードを参 照できるように PROPERTY 指定を付加して おく.
抽出元: UI, クラス・メソッド				
(2-3)	画面表示 メソッド	インスタンス・メソッド		

【図 39】

番号	抽出する要素	変換先の位置	構造、構文上の変換	この抽出・変換の理由づけ
抽出元: UI, クラス見出し部				
(2-1)	クラス名	クラス名 環境部	Controller クラスの接辞を付加 ・ View クラスの接辞を付加 ・ 外部名は命名規則に従い View クラスのファイル名を挿入 ・ Model クラスの接辞を付加 ・ 外部名は命名規則に従い Model クラス のファイル名を挿入	
		View クラスに対するリポート 指定の内部名 Model クラスに対するリポジ トリ指定の内部名		
		クラス終わり見出し	Controller クラスの接辞を付加	
抽出元: UI, クラス変数				
(2-4)	UI メイン・メ ソッド→手続 き部	インスタンス・メソッド→UI メイン・メソッド→手続き部	a. 画面表示メソッドへの INVOKE 文 呼び出し先クラス指定: View クラスへの参照項目の名前に 変える b. チェックメソッドへの INVOKE 文 呼び出し先クラス指定: Model クラスへの参照項目の名前に 変える	画面表示メソッドは View クラス, チェックメソッドは Model クラス のインスタンスを持つようになる ので、呼び出し先を変える。

【図 40】

番号	抽出する要素	変換先の位置	構造, 構文上の変換	この抽出・変換の理由づけ
抽出元: UI, クラス見出し部				
(2-1)	クラス名	クラス名	Model クラスの接辞を付加	
		環境部	・ View クラスの接辞を付加	
		View クラスに対するリポジトリ指定の内部名	・ 外部名は命名規則に従い View クラスのファイル名を挿入	
		Session クラスに対するリポジトリ指定の内部名	・ Session クラスの接辞を付加	
		クラス終わり見出し	・ 外部名は命名規則に従い Session クラスのファイル名を挿入	
			Model クラスの接辞を付加	
抽出元: UI, クラス変数				
(2-2)	トランザクション・レコード	インスタンス変数		
抽出元: UI, クラス・メソッド, マッチング更新メソッド → 変換先: Model, インスタンス・メソッド, マッチング更新メソッド				
(2-5, 主処理), (2-6, 変換処理コントロール), (2-8, 変換処理) は抽出してそのまま挿入				
(2-7)	トランザクション入力		View クラス, View データ取得メソッドへの INVOKE 文	Model クラスは, View クラスから入力データを取得してチェックを行う。
(2-9)	チェック済みトランザクション出力		a. Session クラス, 初期化メソッドへの INVOKE 文を生成 b. Session クラス, トランザクションチェック・メソッドへの INVOKE 文	チェック済みトランザクションは, Session クラスに送る。

【図 4 1】

7. ビュークラス(最終プログラム)

ファイル名: ZMVI.CBL

000001 IDENTIFICATION DIVISION.

000002 CLASS-ID. 在庫マスタ修正 view INHERITS CBL-BASE. [←(2-1)]

000003 ENVIRONMENT DIVISION.

000004 CONFIGURATION SECTION.

000005 REPOSITORY.

000006 CLASS CBL-BASE.

000007 CLASS 在庫マスタ修正 controller IS 'ZMCN'. [←(2-1)]

000008

000009 IDENTIFICATION DIVISION.

000010 CLASS-OBJECT.

000011 PROCEDURE DIVISION.

000012*****

000013* クラス・メソッド

000014*****

000015 IDENTIFICATION DIVISION.

000016 METHOD-ID. 'create'.

000017*****

000018 DATA DIVISION.

000019 WORKING-STORAGE SECTION.

000020 01 W-在庫マスタ修正 view OBJECT REFERENCE SELF.

000021 PROCEDURE DIVISION.

000022 INVOKE SELF 'CBL-NEW'

000023 RETURNING W-在庫マスタ修正 view.

000024 INVOKE 在庫マスタ修正 controller 'create'

000025 USING W-在庫マスタ修正 view.

000026 EXIT METHOD.

000027 END METHOD 'create'.

000028 END CLASS-OBJECT.

000029

000030 IDENTIFICATION DIVISION.

000031 OBJECT.

000032 DATA DIVISION.

000033*****

000034* インスタンス変数

000035*****

000036 WORKING-STORAGE SECTION.

000037 01 T1-在庫マスタ修正レコード.

000038 05 T1-処理区分 PIC X(1) PROPERTY NO SET. [←(2-2)]

000039 05 T1-商品コード PIC X(6) PROPERTY NO SET.

000040 05 T1-商品名 PIC X(30) PROPERTY NO SET.

000041 05 T1-現在庫量 PIC 9(7) PROPERTY NO SET.

000042 05 T1-引当量 PIC 9(7) PROPERTY NO SET.

000043 05 T1-発注点 PIC 9(7) PROPERTY NO SET.

000044 05 T1-基準在庫量 PIC 9(7) PROPERTY NO SET.

000045 05 T1-仕入先コード PIC X(5) PROPERTY NO SET.

000046 01 終了フラグ PIC X(1) PROPERTY NO SET.

⋮

【図 4 2】

(7.ビュークラスの続き)

```
000047 PROCEDURE DIVISION.
000048*****
000049* インスタンス・メソッド
000050*****
000051 IDENTIFICATION DIVISION.
000052 METHOD-ID. 'displayScreen'.
000053*****
000054 DATA DIVISION.
000055 SCREEN SECTION.
000056 01 在庫マスタ細目.
000057     05 LINE 3 COLUMN 10 VALUE '処理区分 : ____'.
000058     05 LINE 3 COLUMN 24 PIC X(1) TO T1-処理区分.
000059     05 LINE 4 COLUMN 10 VALUE '商品コード : ____'.
000060     05 LINE 4 COLUMN 24 PIC X(6) TO T1-商品コード.
000061     05 LINE 5 COLUMN 10 VALUE '商品名 : ____'.
000062     05 LINE 5 COLUMN 24 PIC X(30) TO T1-商品名.
000063     05 LINE 6 COLUMN 10 VALUE '現在庫量 : ____'.
000064     05 LINE 6 COLUMN 24 PIC 9(7) TO T1-現在庫量.
000065     05 LINE 7 COLUMN 10 VALUE '引当量 : ____'.
000066     05 LINE 7 COLUMN 24 PIC 9(7) TO T1-引当量.
000067     05 LINE 8 COLUMN 10 VALUE '発注点 : ____'.
000068     05 LINE 8 COLUMN 24 PIC 9(7) TO T1-発注点.
000069     05 LINE 9 COLUMN 10 VALUE '基準在庫量 : ____'.
000070     05 LINE 9 COLUMN 24 PIC 9(7) TO T1-基準在庫量.
000071     05 LINE 10 COLUMN 10 VALUE '仕入先コード: ____'.
000072     05 LINE 10 COLUMN 24 PIC X(5) TO T1-仕入先コード.
000073     05 LINE 11 COLUMN 40 VALUE '終了 : ____'.
000074     05 LINE 11 COLUMN 24 PIC X(1) TO 終了フラグ.
000075 PROCEDURE DIVISION.
000076     DISPLAY 在庫マスタ細目.
000077     ACCEPT 在庫マスタ細目.
000078     EXIT METHOD.
000079 END METHOD. 'displayScreen'.
000080 END OBJECT.
000081 END CLASS 在庫マスタ修正 view.
```

← (2-3)

← (2-1)

【図 4 3】

8.制御クラス(最終プログラム)

```

ファイル名:ZMCN.CBL
000001 IDENTIFICATION DIVISION.
000002 CLASS-ID. 在庫マスタ修正 controller INHERITS CBL-BASE.  ←(2-1)
000003 ENVIRONMENT DIVISION.
000004 CONFIGURATION SECTION.
000005 REPOSITORY.
000006 CLASS CBL-BASE.
000007 CLASS 在庫マスタ修正 view IS 'ZMV'.  ←(2-1)
000008 CLASS 在庫マスタ修正 model IS 'ZMMD'.
000009
000010 IDENTIFICATION DIVISION.
000011 CLASS-OBJECT.
000012 PROCEDURE DIVISION.
000013 *****
000014* クラス・メソッド
000015*****
000016 IDENTIFICATION DIVISION.
000017 METHOD-ID. 'create'.
000018*****
000019 DATA DIVISION.
000020 LINKAGE SECTION.
000021 01 L-在庫マスタ修正 view OBJECT REFERENCE 在庫マスタ修正 view ONLY.
000022 WORKING-STORAGE SECTION.
000023 01 W-在庫マスタ修正 controller OBJECT REFERENCE SELF.
000024 PROCEDURE DIVISION USING L-在庫マスタ修正 view.
000025 INVOKE SELF 'CBL-NEW'
000026 RETURNING W-在庫マスタ修正 controller.
000027 SET O-在庫マスタ修正 view OF W-在庫マスタ修正 controller
000028 TO L-在庫マスタ修正 view.
000029 INVOKE 在庫マスタ修正 model 'create'
000030 USING O-在庫マスタ修正 view OF W-在庫マスタ修正 controller
000031 RETURNING O-在庫マスタ修正 model OF W-在庫マスタ修正 controller.
000032 INVOKE W-在庫マスタ修正 controller 'uiMain'.
000033 EXIT METHOD.
000034 END METHOD 'create'.
000035 END CLASS-OBJECT.
000036
000037 IDENTIFICATION DIVISION.
000038 OBJECT.
000039 DATA DIVISION.
000040*****
000041* インスタンス変数
000042*****
000043 WORKING-STORAGE SECTION.
000044 01 O-在庫マスタ修正 controller.
000045 05 O-在庫マスタ修正 view OBJECT REFERENCE 在庫マスタ修正 view ONLY PROPERTY.
000046 05 O-在庫マスタ修正 model OBJECT REFERENCE 在庫マスタ修正 model ONLY PROPERTY.
000047
000048 PROCEDURE DIVISION.
000049*****
000050* インスタンス・メソッド
000051*****  ←(2-4)
000052 IDENTIFICATION DIVISION.
000053 METHOD-ID. 'uiMain'.
000054*****
000055 PROCEDURE DIVISION.
000056 PERFORM UNTIL (終了フラグ OF O-在庫マスタ修正 view = '1')
000057 INVOKE O-在庫マスタ修正 view 'displayScreen'
000058 INVOKE O-在庫マスタ修正 model 'changeModel'
000059 END-PERFORM.
000060 EXIT METHOD.
000061 END METHOD 'uiMain'.
000062 END OBJECT.
000063 END CLASS 在庫マスタ修正 controller.  ←(2-1)

```

【図 4 4】

9. モデルクラス(最終プログラム)

ファイル名: ZMMD.CBL

000001 IDENTIFICATION DIVISION.

000002 CLASS-ID. 在庫マスタ修正 model INHERITS CBL-BASE. I←(2-1)

000003 ENVIRONMENT DIVISION.

000004 CONFIGURATION SECTION.

000005 REPOSITORY.

000006 CLASS CBL-BASE.

000007 CLASS 在庫マスタ修正 view IS 'ZMVI'. I←(2-1)

000008 CLASS 在庫マスタ修正 session IS 'ZMSS'.

000009

000010 IDENTIFICATION DIVISION.

000011 CLASS-OBJECT.

000012 PROCEDURE DIVISION.

000013*****

000014* クラス・メソッド

000015*****

000016 IDENTIFICATION DIVISION.

000017 METHOD-ID. 'create'.

000018*****

000019 DATA DIVISION.

000020 LINKAGE SECTION.

000021 01 L-在庫マスタ修正 view OBJECT REFERENCE 在庫マスタ修正 view ONLY.

000022 01 L-在庫マスタ修正 model OBJECT REFERENCE SELF.

000023 WORKING-STORAGE SECTION.

000024 01 W-在庫マスタ修正 model OBJECT REFERENCE SELF.

000025

000026 PROCEDURE DIVISION USING L-在庫マスタ修正 view

000027 RETURNING L-在庫マスタ修正 model.

000028 INVOKE SELF 'CBL-NEW'

000029 RETURNING W-在庫マスタ修正 model.

000030 SET O-在庫マスタ修正 view OF W-在庫マスタ修正 model

000031 TO L-在庫マスタ修正 view.

000032 SET L-在庫マスタ修正 model TO W-在庫マスタ修正 model.

000033 EXIT METHOD.

000034 END METHOD 'create'.

000035 END CLASS-OBJECT.

000036

000037 IDENTIFICATION DIVISION.

000038 OBJECT.

000039 DATA DIVISION.

000040*****

000041* インスタンス変数

000042*****

000043 WORKING-STORAGE SECTION.

000044 01 T1-在庫マスタ修正レコード. I←(2-2)

000045 05 T1-処理区分 PIC X(1).

000046 05 T1-商品コード PIC X(6).

000047 05 T1-商品名 PIC X(30).

000048 05 T1-現在庫量 PIC 9(7).

000049 05 T1-引当量 PIC 9(7).

000050 05 T1-発注点 PIC 9(7).

000051 05 T1-基準在庫量 PIC 9(7).

000052 05 T1-仕入先コード PIC X(5).

000053 01 O-在庫マスタ修正 view OBJECT REFERENCE 在庫マスタ修正 view ONLY.

000054 01 O-在庫マスタ修正 session OBJECT REFERENCE 在庫マスタ修正 session ONLY.

000055

:

【図 4 5】

(9. モデルクラスの続き)

```

000056 PROCEDURE DIVISION.
000057*****
000058* インスタンス・メソッド
000059*****
000060 IDENTIFICATION DIVISION.
000061 METHOD-ID. 'changeModel'.
000062*****
000063 PROCEDURE DIVISION.
000064 主処理.
000065     PERFORM 在庫マスタ入力処理.
000066     PERFORM 変換処理コントロール.
000067     EXIT METHOD.
000068*****
000069 変換処理コントロール.
000070     PERFORM 変換処理.
000071     PERFORM 在庫マスタ入力処理.
000072*****
000073 在庫マスタ入力処理.
000074     INVOKE SELF 'moveFromView'.
000075*****
000076 変換処理.
000077     IF (T1-現在在庫量 IS NUMERIC) AND
000078         (T1-引当量 IS NUMERIC)
000079         THEN
000080             PERFORM チェック済み在庫マスタ修正出力処理
000081         END-IF.
000082*****
000083 チェック済み在庫マスタ修正出力処理.
000084     INVOKE 在庫マスタ修正 session 'create'
000085         RETURNING O-在庫マスタ修正 session.
000086     INVOKE O-在庫マスタ修正 session 'checkTransaction'
000087         USING T1-在庫マスタ修正レコード.
000088 END METHOD 'changeModel'.
000089
000090*****
000091 IDENTIFICATION DIVISION.
000092 METHOD-ID. 'moveFromView'.
000093*****
000094 PROCEDURE DIVISION.
000095     MOVE T1-処理区分 OF O-在庫マスタ修正 view TO T1-処理区分.
000096     MOVE T1-商品コード OF O-在庫マスタ修正 view TO T1-商品コード.
000097     MOVE T1-商品名 OF O-在庫マスタ修正 view TO T1-商品名.
000098     MOVE T1-現在在庫量 OF O-在庫マスタ修正 view TO T1-現在在庫量.
000099     MOVE T1-引当量 OF O-在庫マスタ修正 view TO T1-引当量.
000100     MOVE T1-発注点 OF O-在庫マスタ修正 view TO T1-発注点.
000101     MOVE T1-基準在庫量 OF O-在庫マスタ修正 view TO T1-基準在庫量.
000102     MOVE T1-仕入先コード OF O-在庫マスタ修正 view TO T1-仕入先コード.
000103     EXIT METHOD.
000104 END METHOD 'moveFromView'.
000105 END OBJECT.
000106 END CLASS 在庫マスタ修正 model.

```

← (2-5)

← (2-6)

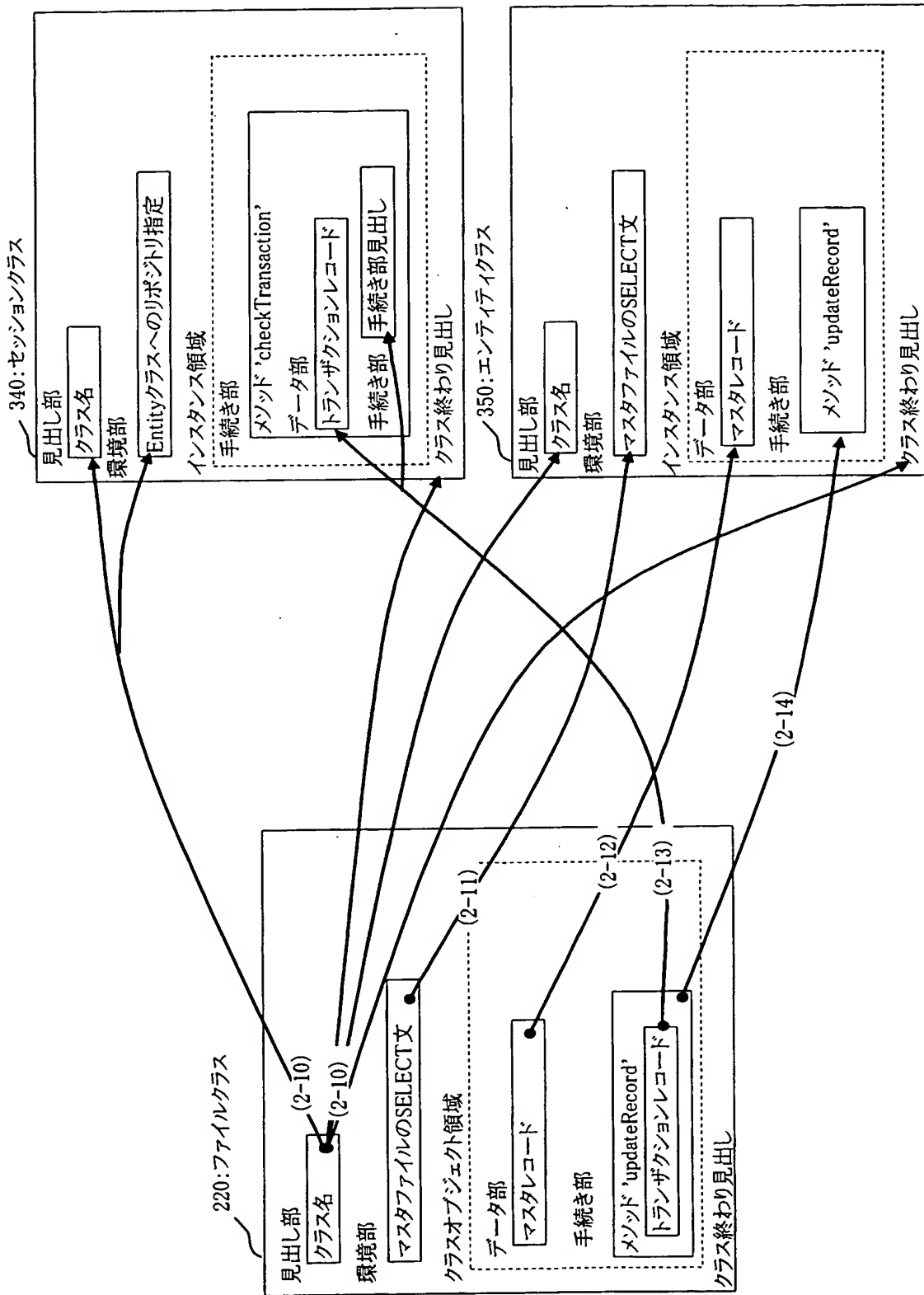
← (2-7)

← (2-8)

← (2-9)

← (2-1)

【図 46】



【図 4 7】

番号	抽出する要素	変換先の位置	構造, 構文上の変換	この抽出・変換の理由づけ
抽出元: FILE, クラス見出し部				
(2-10)	クラス名	クラス名	Session クラスの接辞を付加	
		環境部	・ Entity クラスの接辞を付加	
		Entity クラスに対するリポ ジトリ指定の内部名	・ 外部名は命名規則に従い Entity クラスのファイル名を挿入	
		クラス終わりに見出し	Session クラスの接辞を付加	
抽出元: FILE, マッチング更新メソッド → 変換先: Session, インスタンス・メソッド, トランザクションチェック・メソッド				
(2-13)	トランザク ション・レコ ード	データ部→LINKAGE 節		
		手続き部見出し	レコード名のみ挿入	

【図 48】

番号	抽出する要素	変換先の位置	構造, 構文上の変換	この抽出・変換の理由づけ
抽出元: FILE, クラス見出し部				
(2-10)	クラス名	クラス名	Entity クラスの接辞を付加	
		クラス終わり見出し	//	
抽出元: FILE, 環境部				
(2-11)	マスタ・ファイ ルの SELECT 文	インスタンスの環境部	なし	
抽出元: FILE, クラス変数				
(2-12)	マスタ・レコー ドの定義	インスタンス変数		
抽出元: FILE, マッチング更新メソッド → 変換先: Entity, インスタンス・メソッド, マッチング更新メソッド (2-14, マッチング更新メソッド)は, 抽出してそのまま挿入				

【図 4 9】

10.セッションクラス(最終プログラム)

ファイル名: ZMSS.CBL

000001 IDENTIFICATION DIVISION.

000002 CLASS-ID. 在庫マスタ修正 session INHERITS CBL-BASE. I←(2-10)

000003 ENVIRONMENT DIVISION.

000004 CONFIGURATION SECTION.

000005 REPOSITORY.

000006 CLASS CBL-BASE.

000007 CLASS 在庫マスタ修正 entity IS 'ZMEN'. I←(2-10)

000009 IDENTIFICATION DIVISION.

000010 CLASS-OBJECT.

000011 PROCEDURE DIVISION.

000012*****

000013* クラス・メソッド

000014*****

000015 IDENTIFICATION DIVISION.

000016 METHOD-ID. 'create'.

000017 DATA DIVISION.

000018 LINKAGE SECTION.

000019 01 L-在庫マスタ修正 session OBJECT REFERENCE SELF.

000020 WORKING-STORAGE SECTION.

000021 01 W-在庫マスタ修正 session OBJECT REFERENCE SELF.

000022 PROCEDURE DIVISION RETURNING L-在庫マスタ修正 session.

000023 INVOKE SELF 'CBL-NEW'

000024 RETURNING W-在庫マスタ修正 session.

000025 INVOKE 在庫マスタ修正 entity 'create'

000026 RETURNING O-在庫マスタ修正 entity OF W-在庫マスタ修正 session.

000027 SET L-在庫マスタ修正 session TO W-在庫マスタ修正 session.

000028 EXIT METHOD.

000029 END METHOD 'create'.

000030 END CLASS-OBJECT.

000031

000032 IDENTIFICATION DIVISION.

000033 OBJECT.

000034 DATA DIVISION.

000035*****

000036* インスタンス変数

000037*****

000038 WORKING-STORAGE SECTION.

000039 01 O-在庫マスタ修正 entity OBJECT REFERENCE 在庫マスタ修正 entity ONLY.

000040

⋮

【図 50】

(10.セッションクラスの続き)

```

000041 PROCEDURE DIVISION.
000042*****
000043* インスタンス・メソッド
000044*****
000045 IDENTIFICATION DIVISION.
000046 METHOD-ID. 'checkTransaction'.
000047 DATA DIVISION.
000048 LINKAGE SECTION.
000049 01  T1-在庫マスタ修正レコード.
000050      05  T1-処理区分      PIC X(1).
000051      88  T1-追加 VALUE '1'.
000052      88  T1-更新 VALUE '2'.
000053      88  T1-削除 VALUE '9'.
000054      05  T1-商品コード   PIC X(6).
000055      05  T1-商品名       PIC X(30).
000056      05  T1-現在庫量    PIC 9(7).
000057      05  T1-引当量      PIC 9(7).
000058      05  T1-発注点     PIC 9(7).
000059      05  T1-基準在庫量 PIC 9(7).
000060      05  T1-仕入先コード PIC X(5).
000061 WORKING-STORAGE SECTION.
000062 01  W-ファイルエラーフラグ PIC X(1).
000063      88 レコードあり VALUE '1'.
000064      88 レコードなし VALUE '0'.
000065
000066 PROCEDURE DIVISION USING T1-在庫マスタ修正レコード
000067      INVOKE O-在庫マスタ修正 entity 'recordExists'
000068      RETURNING W-ファイルエラーフラグ.
000069      EVALUATE T1-処理区分
000070      WHEN T1-追加
000071      IF レコードなし
000072      THEN
000073          INVOKE O-在庫マスタ修正 entity 'updateRecord'
000074          USING T1-在庫マスタ修正レコード
000075      END-IF
000076      WHEN T1-更新 T1-削除
000077      IF レコードあり
000078      THEN
000079          INVOKE O-在庫マスタ修正 entity 'updateRecord'
000080          USING T1-在庫マスタ修正レコード
000081      END-IF
000082      END-EVALUATE.
000083      EXIT METHOD.
000084 END METHOD 'checkTransaction'.
000085 END OBJECT.
000086 END CLASS 在庫マスタ修正 session.

```

← (2-13)

← (2-13)

← (2-10)

【図 5 1】

11. エンティティクラス(最終プログラム)

ファイル名: ZMEN.CBL

000001 IDENTIFICATION DIVISION.

000002 CLASS-ID. 在庫マスタ修正 entity INHERITS CBL-BASE. [← (2-10)

000003 ENVIRONMENT DIVISION.

000004 CONFIGURATION SECTION.

000005 REPOSITORY.

000006 CLASS CBL-BASE.

000007

000008 IDENTIFICATION DIVISION.

000009 CLASS-OBJECT.

000010 PROCEDURE DIVISION.

000011]*****

000012* クラス・メソッド

000013*****

000014 IDENTIFICATION DIVISION.

000015 METHOD-ID. 'create'.

000016 DATA DIVISION.

000017 LINKAGE SECTION.

000018 01 L-在庫マスタ修正 entity OBJECT REFERENCE SELF.

000019 WORKING-STORAGE SECTION.

000020 01 W-在庫マスタ修正 entity OBJECT REFERENCE SELF.

000021 PROCEDURE DIVISION RETURNING L-在庫マスタ修正 entity.

000022 INVOKE SELF 'CBL-NEW'

000023 RETURNING W-在庫マスタ修正 entity.

000024 SET L-在庫マスタ修正 entity TO W-在庫マスタ修正 entity.

000025 EXIT METHOD.

000026 END METHOD 'create'.

000027 END CLASS-OBJECT.

000028

000029 IDENTIFICATION DIVISION.

000030 OBJECT.

000031 ENVIRONMENT DIVISION.

000032 INPUT-OUTPUT SECTION.

000033 FILE-CONTROL.

000034 SELECT M1-在庫マスタファイル

000035 ASSIGN TO "SYS030-DA-DK-1"

000036 ORGANIZATION INDEXED

000037 ACCESS RANDOM

000038 RECORD KEY M1-商品コード.

000039 DATA DIVISION.

000040*****

000041* インスタンス変数

000042*****

000043 FILE SECTION.

000044 FD M1-在庫マスタファイル.

000045 01 M1-在庫マスタレコード.

000046 05 M1-商品コード PIC X(6).

000047 05 M1-商品名 PIC X(30).

000048 05 M1-現在庫量 PIC S9(7).

000049 05 M1-引当量 PIC S9(7).

000050 05 M1-発注点 PIC S9(7).

000051 05 M1-基準在庫量 PIC S9(7).

000052 05 M1-仕入先コード PIC X(5).

000053

⋮

【図 5 2】

(11.エンティティクラスの続き(1))

```
000054 PROCEDURE DIVISION.
000055 *****
000056 * インスタンス・メソッド
000057 *****
000058 IDENTIFICATION DIVISION.
000059 METHOD-ID. 'recordExists'.
000060 *****
000061 DATA DIVISION.
000062 LINKAGE SECTION.
000063 01 L-商品コード PIC X(6).
000064 01 L-ファイルエラーフラグ PIC X(1).
000065     88 レコードあり VALUE '1'.
000066     88 レコードなし VALUE '0'.
000067 PROCEDURE DIVISION USING L-商品コード
000068     RETURNING L-ファイルエラーフラグ
000069 OPEN INPUT M1-在庫マスタファイル
000070 MOVE L-商品コード TO M1-商品コード.
000071 READ M1-在庫マスタファイル
000072     INVALID
000073         SET レコードなし TO TRUE
000074     NOT INVALID
000075         SET レコードあり TO TRUE
000076     END-READ.
000077     EXIT METHOD.
000078 END METHOD 'recordExists'.
000079 *****
000080 *****
000081 IDENTIFICATION DIVISION.
000082 METHOD-ID. 'updateRecord'.
000083 *****
000084 DATA DIVISION.
000085 LINKAGE SECTION.
000086 01 T1-在庫マスタ修正レコード.
000087     05 T1-処理区分 PIC X(1).
000088         88 T1-追加 VALUE '1'.
000089         88 T1-更新 VALUE '2'.
000090         88 T1-削除 VALUE '9'.
000091     05 T1-商品コード PIC X(6).
000092     05 T1-商品名 PIC X(30).
000093     05 T1-現在在庫量 PIC 9(7).
000094     05 T1-引当量 PIC 9(7).
000095     05 T1-発注点 PIC 9(7).
000096     05 T1-基準在庫量 PIC 9(7).
000097     05 T1-仕入先コード PIC X(5).
000098 *****
000099 WORKING-STORAGE SECTION.
000100 01 作業領域.
000101     05 W-トランザクションキー PIC X(6) VALUE LOW-VALUE.
000102     05 W-マスタキー PIC X(6).
000103 01 フラグ.
000104     05 ファイル終了フラグ1 PIC X VALUE '0'.
000105     05 ファイル終了フラグ2 PIC X VALUE '0'.
000106     05 処理終了フラグ PIC X VALUE '0'.
000107 *****
000108 PROCEDURE DIVISION USING T1-在庫マスタ修正レコード.
000109 主処理.
000110 OPEN I-O M1-在庫マスタファイル
000111 PERFORM 在庫マスタ修正入力処理.
000112 PERFORM 旧在庫マスタ入力処理.
000113 PERFORM 更新コントロール処理.
000114 CLOSE M1-在庫マスタファイル
000115 EXIT METHOD.
000116 *****
000117 更新コントロール処理.
000118 IF (W-トランザクションキー = W-マスタキー)
000119     THEN
000120         PERFORM 合致処理
000121     ELSE
000122         PERFORM トランザクション処理
000123 END-IF.
```

← (2-14)

⋮

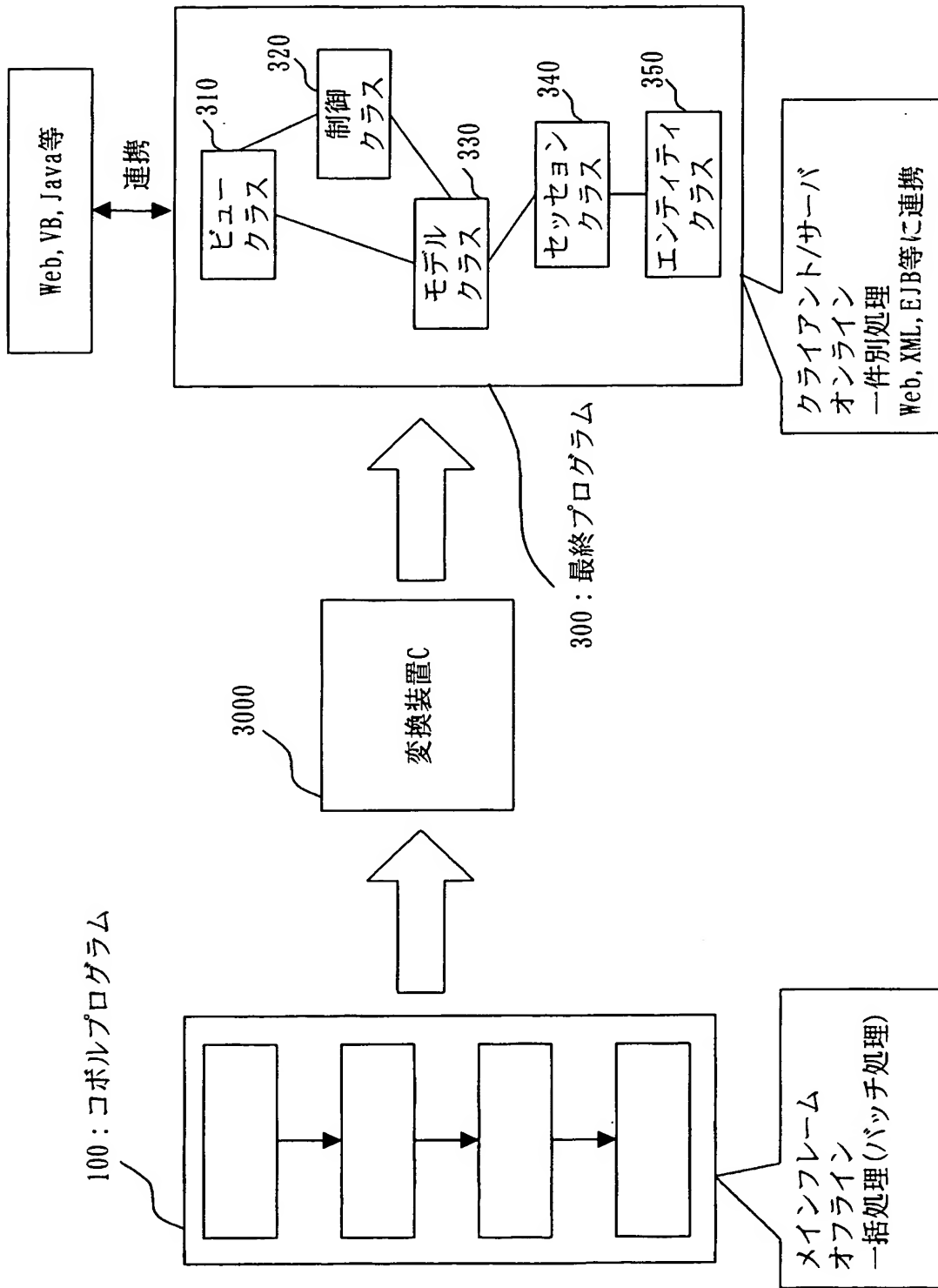
【図 5 3】

(11. エンティティクラスの続き(2))

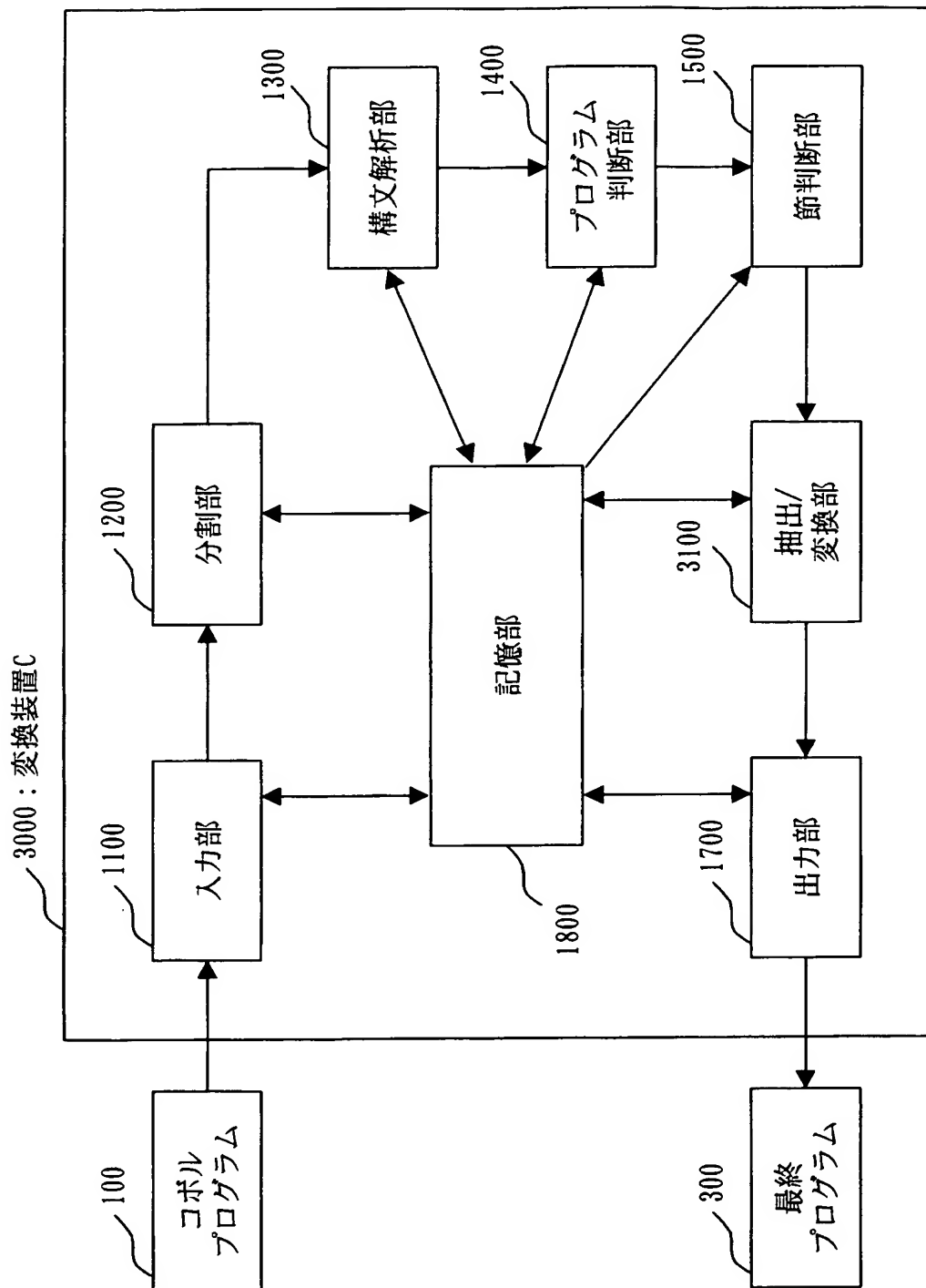
```
000125*****
000126  合致処理.
000127      IF (W-トランザクションキー = HIGH-VALUE)
000128          THEN
000129              MOVE '1' TO 処理終了フラグ
000130          ELSE
000131              EVALUATE T1-処理区分
000132              WHEN T1-更新
000133                  PERFORM 更新処理
000134                  PERFORM 新在庫マスタ出力処理
000135              WHEN T1-削除
000136                  PERFORM 削除処理
000137              END-EVALUATE
000138          END-IF.
000139*****
000140  トランザクション処理.
000141      IF T1-追加
000142          THEN
000143              PERFORM 追加処理
000144              PERFORM 追加新在庫マスタ出力処理.
000145          END-IF.
000146*****
000147  更新処理.
000148      MOVE T1-商品名          TO M1-商品名.
000149      MOVE T1-現在在庫量      TO M1-現在在庫量.
000150      MOVE T1-引当量          TO M1-引当量.
000151      MOVE T1-発注点          TO M1-発注点.
000152      MOVE T1-基準在庫量      TO M1-基準在庫量.
000153      MOVE T1-仕入先コード    TO M1-仕入先コード.
000154*****
000155  追加処理.
000156      MOVE T1-商品コード      TO M1-商品コード.
000157      MOVE T1-商品名          TO M1-商品名.
000158      MOVE T1-現在在庫量      TO M1-現在在庫量.
000159      MOVE T1-引当量          TO M1-引当量.
000160      MOVE T1-発注点          TO M1-発注点.
000161      MOVE T1-基準在庫量      TO M1-基準在庫量.
000162      MOVE T1-仕入先コード    TO M1-仕入先コード.
000163*****
000164  追加新在庫マスタ出力処理.
000165      WRITE M1-在庫マスタレコード
000166      INVALID MOVE '1' TO 処理終了フラグ.
000167*****
000168  新在庫マスタ出力処理.
000169      REWRITE M1-在庫マスタレコード
000170      INVALID MOVE '1' TO 処理終了フラグ.
000171*****
000172  削除処理.
000173      DELETE M1-在庫マスタレコード
000174      INVALID MOVE '1' TO 処理終了フラグ.
000175*****
000176  在庫マスタ修正入力処理.
000177      MOVE T1-商品コード      TO W-トランザクションキー M1-商品コード
000178      TO W-トランザクションキー M1-商品コード
000179*****
000180  旧在庫マスタ入力処理.
000181      READ M1-在庫マスタファイル
000182      INVALID MOVE '1' TO ファイル終了フラグ2.
000183      IF (ファイル終了フラグ2 = '1')
000184          THEN
000185              MOVE HIGH-VALUE TO W-マスタキー
000186          ELSE
000187              MOVE M1-商品コード TO W-マスタキー
000188          END-IF.
000189  END METHOD 'updateRecord'.
000190  END OBJECT.
000191  END CLASS 在庫マスタ修正 entity. I←(2-10)
```

←(2-14
つづき)

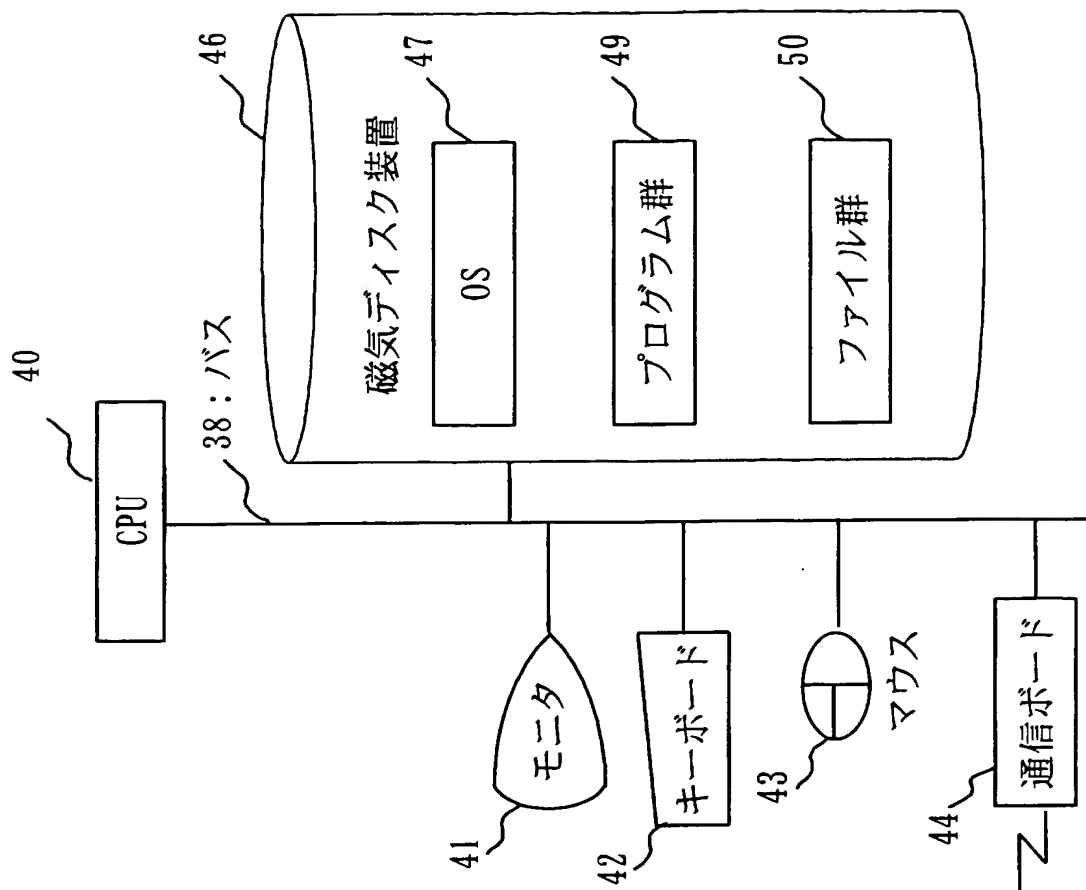
【図 5 4】



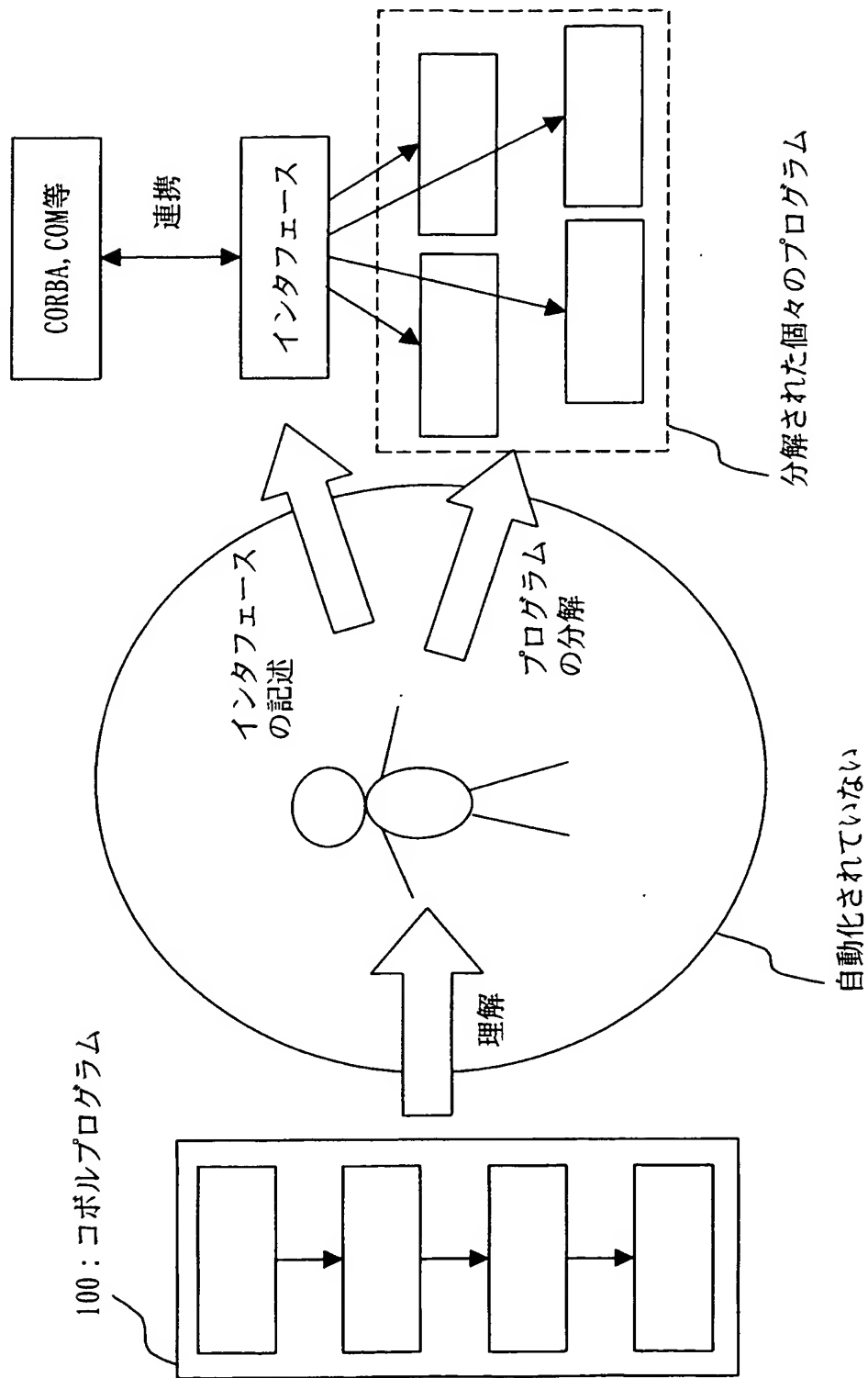
【図 55】



【図 56】



【図 57】



【書類名】 要約書

【要約】

【課題】 本発明は、既存のプログラムを新しい技術のソフトウェアに適した構造に変換することを目的とする。

【解決手段】 入力部 1100 はコボルプログラム 100 を入力し、記憶部 1800 に記憶する。分割部 1200 はコボルプログラム 100 をまとまりある複数のプログラムに分解し、構文解析部 1300 は分解されたそれぞれのプログラムの構文を解析する。プログラム判断部 1400 は各プログラムの役割を判断し、節判断部 1500 は各プログラム中の節の内容、役割を判断する。これらの動作を終えた後、抽出／変換部 3100 は最終プログラム 300 を生成するためのデータの抽出及び変換を行ない、その結果、コボルプログラム 100 のソースコードから最終プログラム 300 のソースコードへの変換が自動的に行われる変換装置を提供する。

【選択図】 図 1

【書類名】 手続補正書
【整理番号】 000000345
【提出日】 平成15年 9月10日
【あて先】 特許庁長官殿
【事件の表示】
 【出願番号】 特願2002-278362
【補正をする者】
 【識別番号】 899000079
 【氏名又は名称】 学校法人 慶應義塾
【代理人】
 【識別番号】 100099461
 【弁理士】
 【氏名又は名称】 溝井 章司
【手続補正1】
 【補正対象書類名】 特許願
 【補正対象項目名】 発明者
 【補正方法】 変更
 【補正の内容】
 【発明者】
 【住所又は居所】 神奈川県横浜市港北区日吉三丁目14番1号 慶應義塾大学理工学部内
 【氏名】 永田 守男
 【発明者】
 【住所又は居所】 神奈川県逗子市池子3丁目13番28号
 【氏名】 魚田 勝臣
 【発明者】
 【住所又は居所】 神奈川県横浜市港北区日吉三丁目14番1号 慶應義塾大学理工学部内
 【氏名】 梶尾 義規
【その他】 [発明者]の [住所又は居所]で「神奈川県横浜市港北区日吉三丁目14番1号 慶應義塾大学理工学部内」とすべきところを、「神奈川県横浜市港北区日吉3丁目14番 慶應義塾大学理工学部内」と誤って記載したため。

特願 2 0 0 2 - 2 7 8 3 6 2

出 願 人 履 歴 情 報

識別番号

[8 9 9 0 0 0 0 7 9]

1. 変更年月日

1 9 9 9 年 9 月 1 7 日

[変更理由]

新規登録

住 所

東京都港区三田2丁目15番45号

氏 名

学校法人慶應義塾